Helsinki University of Technology
Department of Electrical and Communications Engineering
Networking Laboratory

Mikko Uljas

Monitoring and System Management in Distributed Environment

This thesis has been submitted in partial fulfillment of the requirements for the degree of Master of Science in computer science and engineering.

Tekijä:	Mikko Antero Uljas
Työn nimi:	Tietojärjestelmien Valvonta ja Hallinta Hajautetussa Ympäristössä
Päivämäärä:	20. helmikuuta, 2004
Sivumäärä:	84
Osasto:	Sähkö- ja tietoliikennetekniikan osasto
Professuuri:	S-38 Tietoverkkotekniikka
Työn valvoja:	Professori Jorma Jormakka
Työn ohjaaja:	Tommi Förbom, FM
Tiivistelmäteksti:	Organisaatioiden tietojärjestelmien muuttuessa yhä monimutkaisemmiksi, hajautettujen järjestelmien yleistyessä ja järjestelmien välisen integraation tullessa yhä yleisemmäksi kohtaa kokonaisuuden hallinta uusia haasteita. Samaan aikaan erilaiset liikkeenjohdolliset konseptit tarvitsevat tietoa järjestelmien käyttäytymisestä pystyäkseen seuraamaan kuinka asetettu tavoitteet toteutuvat. Tässä työssä esittelemme kolmitasoisen kehyksen hajautettujen tietojärjestelmien hallintaan ja arvioimme sen sopivuutta yrityksen tietojärjestelmien valvonta-arkkitehtuuriksi. Oikein toteutettu valvontajärjestelmä ei pelkästään mahdollista järjestelmien sen hetkisen tilan valvontaa vaan myös tukee liikkeenjohdollisten konseptien tietotarpeita. Hajautettujen järjestelmien läpikohtainen valvonta tuottaa valtavan määrän dataa. Arvioimme keskitettyä tietovarasto-ratkaisua tapana minimoida valvontajärjestelmän tuottama kuormitus itse valvottavalle järjestelmälle ja tarjota keskitetty paikka päästä tietoon käsiksi jatkoanalysointia varten. Lopuksi käsittelemme liikkeenjohdollisia konsepteja ja osoitamme minkälaisia hyötyjä oikein toteutettu valvontajärjestelmä voi niille tarjota
Avainsanat:	Valvontajärjestelmä, hajautettu järjestelmä, tietojärjestelmän hallinta, keskitetty tietovarasto, hallinta-agentti

Author:	Mikko Antero Uljas
Name of the thesis:	Monitoring and System Management in Distributed Environment
Date:	February 20, 2004
Number of pages:	84
Department:	Department of Electrical and Communications Engineering
Professorship	S-38 Networking Technology
Supervisor:	Professori Jorma Jormakka
Instructor:	Tommi Förbom, M.Sc.
Abstract:	As organizations information systems get more complex, distributed systems more general and integration between systems increasingly popular also management of the whole meets new challenges. At the same time different management concepts need information of systems behavior to be able to follow how agreed goals are accomplished and how agreed service levels are met. In this thesis we will present a three-tier framework for distributed system management and consider its suitability for enterprise level architecture. Properly implemented monitoring system not only enables the possibility to identify system's current state but also supports management concepts information needs. Throughout monitoring of distributed systems generates massive amount of data. A data warehouse solution is considered as a way to minimize monitoring overhead and to provide a centralized location where the data can be accessed for further analysis. At the end we will address different management concepts and show what they can gain from properly implemented monitoring system.
Keywords:	Monitoring system, distributed system, system management, data warehouse, management agent

Preface

Time flies when you have fun. The need to write the thesis came in front of me too fast. Not because it is not high time for me to graduate, but because it also means I have to leave carefree student life behind. Middle age crisis, burnout and all other adult diseases - welcome.

I would like to thank my instructor Tommi Förbom and my supervisor Jorma Jormakka for the feedback and encouraging comments during the process. Also special thanks to my brother Matti for borrowing his headphones. I would also like to thank my family and friends for encouragement and support during my studies

Mikko Uljas

Helsinki, February 20, 2004

Table of Contents

List of Figures

List of Tables

Acronyms

CCTA	Central Computer and Telecommunications Agency
CMDB	Capacity Management Database
CMIP	Common Management Information Protocol
CPU	Central Processing Unit
DCE	Distributed Computing Environment
DNS	Domain Name Service
ER	Entity-Relationship model
GGF	Global Grid Forum
IP	Internet Protocol
ISP	Internet Service Provider
ITIL	IT Infrastructure Library
I/O	Input/Output
KM	Knowledge Module
MDW	Measurement Data Warehouse
MIR	Management Information Repository
MTBF	Mean Time Between Failure
MTTR	Mean Time To Repair
OLA	Operational Level Agreement
OLAP	Online Analytical Processing
OLTP	Online Transaction Processing
OSF	Open Software Foundation
OSI	Open Systems Interconnection
PMI	Control-M Integration Module for Patrol
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RDMS	Relational Database Management System
SQL	Structured Query Language
SLA	Service Level Agreement
SLM	Service Level Management
SNMP	Simple Network Management Protocol
WWW	World Wide Web

1. INTRODUCTION

As enterprises information systems get more complex, distributed systems more general and integration between systems increasingly popular also management of the whole meets new challenges. System management and monitoring with local solutions or ad-hoc methods is getting more difficult.

M. A. Bauer et al also recognized this when they stated, as organizations move toward distributed computing environments, there will be a corresponding growth in distributed applications central to the enterprise. The design, development, and management of distributed applications present many difficult challenges. As these systems grow to hundreds or even thousands of devices and similar or greater magnitude of software components, it will become increasingly difficult to manage them without appropriate support tools and frameworks. [BAU97]

At the same time different management concepts are getting increasingly popular. Management concepts need information of systems behavior to be able to determine the right metrics and to follow how agreed goals are accomplished. A good example of this kind management concept is Service Level Management defined in common IT Infrastructure Library (ITIL). One needs to collect and interpret performance information from many different systems and processes to follow up how agreed service levels are met.

There is pressure coming from two sides. Information systems are becoming increasingly harder to manage because systems are distributed to many locations and devices. Also relations between systems are getting more complex because of the increased integration. At the same time there is increasing demand of detailed performance information. Customers (internal and external) and company's management want to know how their needs are met.

Creation of a monitoring architecture and monitoring information's collection to a centralized data warehouse can offer tools to manage complex systems systematically. The centralized data warehouse enables monitoring data's further analyzes and creates

a centralized point where other applications can access the data. In addition to other gains management applications can also use the warehouse for their own data storage needs.

1.1. Content of the master thesis

In this thesis we will present a common management framework, consider its suitability for enterprise level monitoring system, introduce a measurement data warehouse solution for monitoring information storing and consider what kind of gains management concepts can get from a properly implemented monitoring system. The aim is to give general idea how distributed systems monitoring can be arranged and what are the gains. Because of the wide subject there is only little room for details. In most cases we will restrict ourselves to merely presenting the ideas gathered from many sources and try to bind them together. In many cases there is need for more theoretical work, prototypes and case studies.

We propose a three-tier solution for the enterprise level monitoring architecture. Agents located in every managed node are responsible for collecting the information and maintaining detailed data in a local data source. A data warehouse solution is responsible for collecting the information to a centralized database. Management applications can access data from the data warehouse. All this is encapsulated in the common management framework.

The common management framework proposed in the thesis binds everything together and joins the monitoring system to a broader distributed system management frame. Although we have concentrated our efforts to monitoring we can not forget the bigger picture. The true power of monitoring is achieved when it is integrated into other management applications.

After presenting the framework and common architecture we turn our focus to the centralized database. We present a proposed Measurement Data Warehouse solution in more detail and show how monitoring information can be archived in there. We also present a monitoring information data model and consider how data should be handled in the warehouse.

Then we address scalability and resource usage issues and try to estimate suitability of our solution to enterprises. Motivation is to show that the solution is suitable for enterprise level systems. Resource usage calculations are based on theoretical values and there is need for more studies.

At the end, we address different management concepts and show what they can gain from a properly implemented monitoring system. Management concepts have different needs considering monitoring information. We try to show what kind of information is valuable for management concepts and how that information can be obtained. We also show how the proposed framework can speed up management applications development and implementation process.

Let us start from the basics. Following chapters will introduce you the basic terminology and framing of question of monitoring.

1.2. Monitoring of distributed systems

Monitoring is the extraction of dynamic information concerning a computational process, as that process executes [SNO88]. The monitoring of distributed systems involves:

- collection
- interpretation (analysis)
- display

of information concerning the interactions among concurrently executing processes [JOY87]. Monitoring information must be collected somehow. This can be done for example by using dedicated agents responsible for collecting information from many sources. Gathered information itself is useless and needs to be interpreted. Certain events can be looked for from the information flow or collected information can be further analyzed in many different ways. Gathered and interpreted information will be displayed either in graphical form for humans or in raw data form for applications.

Depending of type and detail of gathered information and its display it can support:

- debugging
- testing (how application behaves in detail)
- dynamic adaptation
- dynamic documentation of distributed systems
- performance evaluation and capacity management
- different management concepts (for example Service Level Management)
- identifying system's current state (status reporting)
- reconfiguration

Broad scope of different possible uses of monitoring information also brings up one of the basic problems when determining type and detail of monitored information.

During monitoring, resources are shared between monitoring system components and the monitored system. Thus, the monitored system has to maintain its functionality with fewer resources. If monitoring system components consume a great deal of resources, this eventually affects performance of the monitored system. Unwanted effects can range from slow response times of processes to total deadlocks of the system.

The combination of all effects of monitoring on the monitored distributed system is referred to as monitoring overhead. To use the resources in the system efficiently, one has to select a configuration with minimal monitoring overhead [ABD96].

Debugging and testing demand very detailed monitoring information. Also time interval between measurements must be very short (almost continuous). Especially in distributed systems the amount of monitoring information will be enormous. In production environments this will most likely be impossible because of the additional stress to network and system resources.

Capacity management, performance evaluation and other management concepts are examples of monitoring data uses that do not require so detailed information.

Identifying systems current state and dynamic adaptation lie somewhere in between these two end points.

The thesis concentrates on collecting and interpreting monitoring information from the enterprise level. The amount of collected data has to be kept reasonable low. Thus, debugging and testing as a possible uses of data are left out of the scope of the thesis.

We consider a data warehouse system as a way to minimize monitoring overhead. More detailed monitoring information can be archived locally. Only selected and aggregated information is stored in the centrally maintained data warehouse. This will be explained in more detail in chapter 3.

1.3. Background and related studies

One of the first papers that discussed the use of relational database for monitoring was by Snodgrass in 1988 [SNO88]. He developed a relational approach to monitor complex systems.

Monitoring is an essential part of many program development tools, and plays a central role in debugging, optimization, status reporting, and reconfiguration. Traditional monitoring techniques are inadequate when monitoring complex systems such as multiprocessors or distributed systems. A new approach is described in which a historical database forms the conceptual basis for the information processed by the monitor. This approach permits advances in specifying the low-level data collection, specifying the analysis of the collected data, performing the analysis, and displaying the results [SNO88].

These ideas are still valid. Information gathered with monitors is collected to a historical database. The historical database can be organized as a data warehouse to provide more efficient queries. SQL (Structured Query Language) [SQL92] provides general and powerful language for data extracting.

After Snodgrass several researchers have addressed the problem with different scopes and from different points of view. Commercial solutions have also been developed.

Next we will shortly introduce three different studies. Two of them are closely related to the thesis and one has taken a different approach worth mentioning.

1.3.1. Streaming monitoring data collection to a centralized database

In year 2002 J. Lee et al presented [LEE02] a sample system for monitoring data collection and interpretation. They proposed a relational data archive where the monitoring data is stored. Their system has four main components that take care of the data collection part. These components are illustrated in Figure 1:

- application instrumentation, which produces the monitoring data
- monitoring activation service, which collects events and sends them to the requested destination
- monitoring event receiver, which consumes the monitoring data and converts events to SQL records and writes them to a disk buffer
- archive feeder, which loads SQL records into a database

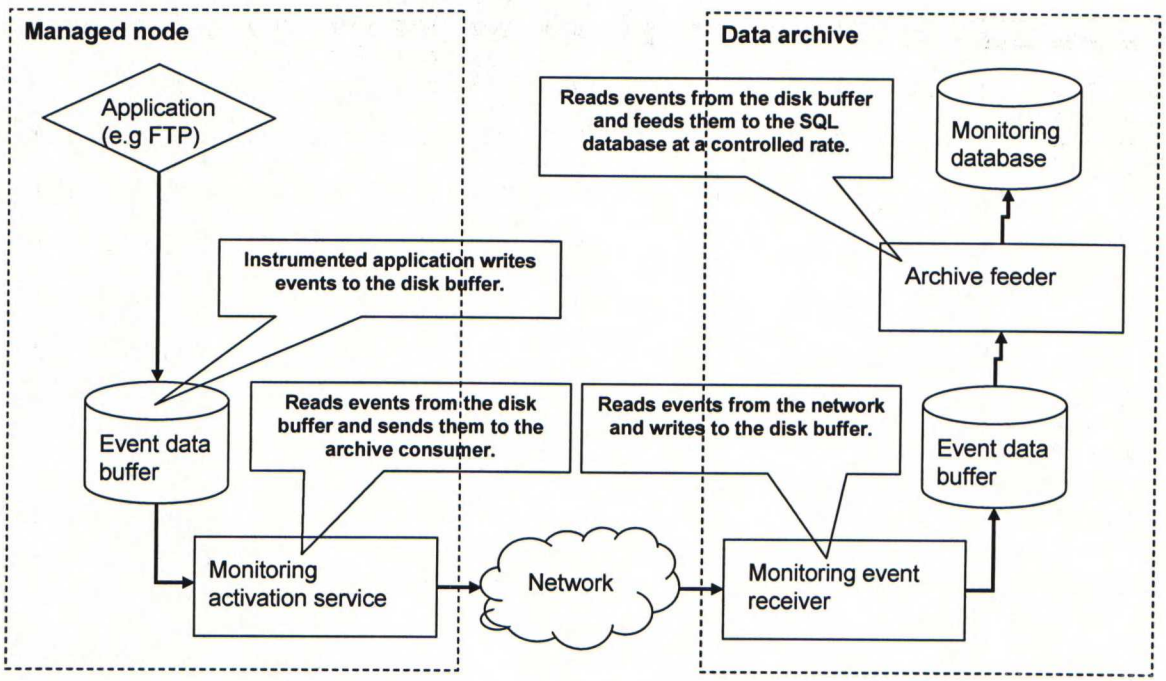

Figure 1: Monitoring data collection components based on [LEE02].

Their approach was to stream collected information immediately into a centralized historical database. Monitoring information interpretation and display are done through an analysis client that gets all data from the centralized database. The prototype system was mainly addressed for performance evaluation of distributed applications, but we have also evaluated method's suitability as an enterprise level monitoring architecture.

1.3.2. Framework for distributed applications management

Few years before (1997) M. A. Bauer et al presented [BAU97] a framework for management of distributed applications. The framework is based on a set of common management services that support management activities. Services include monitoring, control, configuration, and data repository services. The framework is presented in Figure 2.

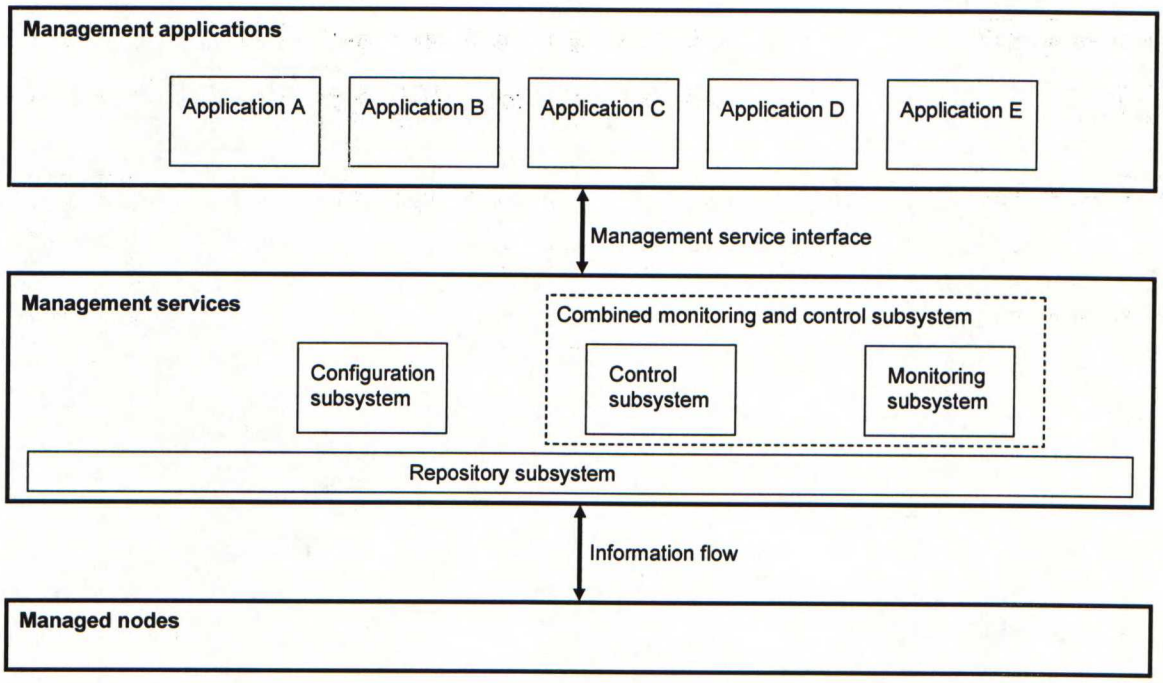

Figure 2: Bauer's distributed applications management framework

The framework was originally addressed for managing distributed applications and systems. Their prototype management system was built on an Open Software Foundation's Distributed Computing Environment (OSF DCE) [WEB3].

In the thesis we have adopted the framework as a common architecture. Monitoring data collection, interpretation and display are considered as a part of the framework. The framework will be presented more detailed in chapter 2.

1.3.3. DNS type approach

Last year (2003) R. Renesse et al presented [REN03] a different approach - information management service called Astrolabe. Like DNS, Astrolabe organizes the resources into a hierarchy of domains and associates attributes with each domain. It monitors the state of information collection of distributed resources and reports summaries of the information to users. Summaries of the data are computed in the system using on-the-fly aggregation. The aggregation mechanism is controlled by SQL queries.

To a user Astrolab looks much like a database, although it is a virtual database that does not reside on a centralized server. In the thesis we have taken a different approach and propose a centralized data warehouse. What Astrolab suggests is a distributed monitoring system that relies on distributed resources.

1.3.4. Miscellaneous and commercial solutions

The Global Grid Forum's[1] Relational Database Information Services research group is also advocating the use of relational models for storing monitoring data and has produced a number of documents, such as [FIS01] and [DIN01].

From commercial system provides we could mention BMC Software [BMC1] and Hewlett-Packard Company (HP) [HP1]. They both offer several kinds of system management products. BMC Software's monitoring solution is called PATROL and

---

[1] The Global Grid Forum (GGF) is a community-initiated forum of 5000+ individual researchers and practitioners working on distributed computing, or "grid" technologies. GGF's primary objective is to promote and support the development, deployment, and implementation of Grid technologies and applications via the creation and documentation of "best practices" - technical specifications, user experiences, and implementation guidelines.

the control solution is called CONTROL-M. HP's monitoring solution is called HP Openview.

2. THE COMMON MANAGEMENT FRAMEWORK

M. A. Bauer et al has presented [BAU97] a framework for management of distributed applications. Their prototype management system was built on an Open Software Foundation's Distributed Computing Environment (OSF DCE). In the thesis the framework is taken as a common architecture. Monitoring data collection, interpretation and display are considered as a part of the framework.

There are several motives behind the decision. The main one is that the framework considers distributed systems management as a whole. There are other papers considering one or two aspects [JOY87, REN03] of distributed systems monitoring, but Bauer provides the common framework.

Its three-tier architecture is very suitable for enterprise level information systems. Its monitoring data collection can be applied to all management areas (network, system and distributed applications management). This is illustrated in Figure 3. The framework also provides an interface to management applications for monitoring information interpretation and display.

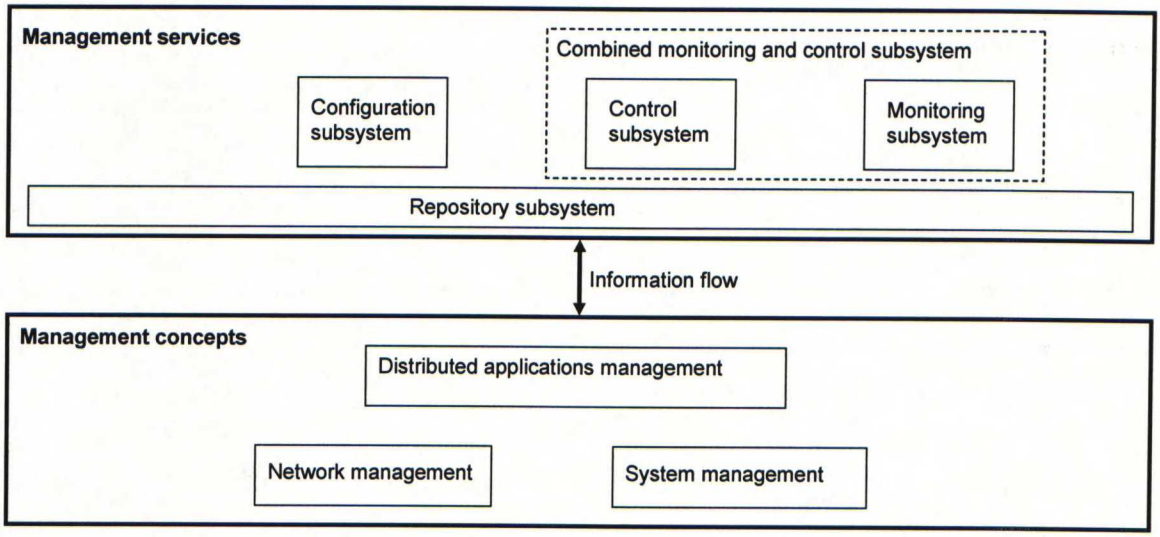

Figure 3: How general management concepts apply to the framework

First Bauer's framework is presented as a whole. Then its essential components are discussed in more detail (especially the ones concerning monitoring). Then possible

different design approaches and considerations are presented concerning in particular the implementation of monitoring data collection, interpretation and display.

2.1. The framework for distributed applications management

The framework is based on a set of common management services that support management activities. Services include combined monitoring and control, configuration, and data repository services. Figure 4 presents the whole framework.

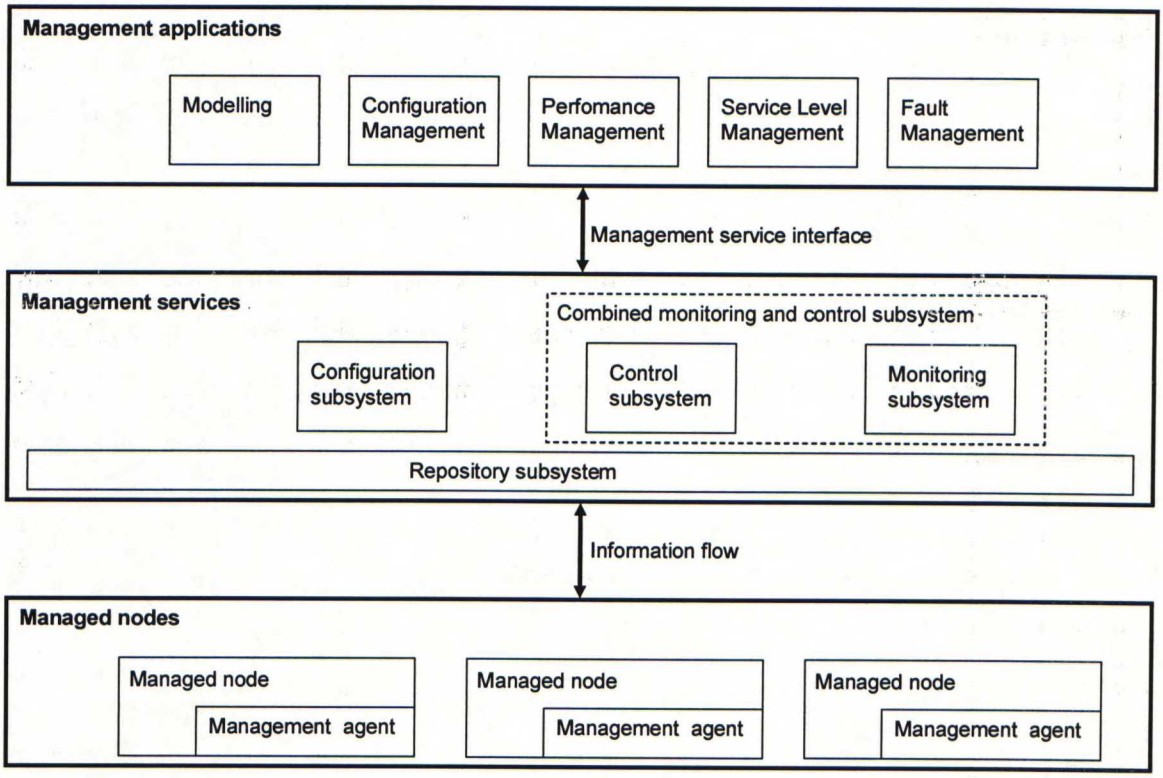

Figure 4: Bauer's distributed applications management framework

2.1.1. Management applications

Management applications are used to perform management tasks, such as system configuration, capacity management, incident management, service level management, report generation, visualization of network or system activity, simulation and modeling. Applications communicate with management services through a common management services interface. A common interface has many advantages and it speeds up management application's development process.

2.1.2. Managed nodes

Managed nodes are abstractions of real managed resources such as servers or network devices. Management agents carry out management activities on behalf of management services and applications. Management agents can collect and store monitoring information, respond to management requests and generate event notifications. Management services may communicate with management agents using SNMP [CAS90], CMIP [ISO1] or proprietary protocols.

2.1.3. Management services

Management services are playing a central role in the framework. Services are organized in three (possible four) subsystems:

- repository services subsystem
- configuration services subsystem
- combined monitoring and control services subsystem
    - monitoring services subsystem
    - control services subsystem

Each of management services subsystems are described in detail in following chapters.

2.1.3.1. Repository services subsystem

The repository subsystem provides database management services needed by management applications and other subsystems. Three types of data must be handled by the repository subsystem: structural data, control data and measurement data. Table 1 describes the characteristics of different types of data.

Type of data	Short description
Structural data	Consists of descriptions of managed objects, their relationships and their environments. These include for example, application configurations, workload

	characteristics and network topology
Control data	Capture information related to the operation of an application and are of two types. The first type is a set of environment and initialization values for an application instance. The second type is a set of event notifications generated by a managed object.
Measurement data	Describe the run-time operation of an object. They may be data collected by monitoring the object, such as process CPU use or process disk I/O, or they may be derived from collected data, such as "resource use = CPU use + disk I/O." The measurement data provides both the current state and the execution history of an object.

Table 1: Three types of data that must be handled by the repository subsystem

Different characteristics of three types of data mean that no single type of database system could efficiently support all the data. This fact led to a repository subsystem composed of two different repositories: a Management Information Repository (MIR) to store structural and control data and a Measurement Data Warehouse (MDW) to store measurement data. Figure 5 presents repository data subsystem in detail.

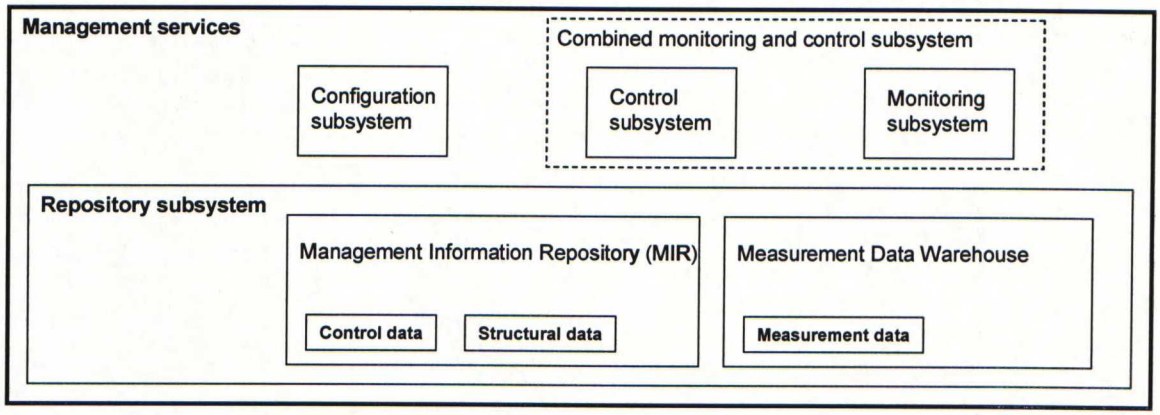

Figure 5: Repository data subsystems in detail

The Management Information Repository implements the information model and is used to store structural data and control data. It can be implemented with a client/server architecture in which the MIR server stores the data and provides retrieval and update functions to clients. Clients can retrieve data from the server in two ways, by issuing queries to the server searching for objects matching certain criteria, or by using a browsing paradigm to locate objects by exploiting the structure of the information model.

The Measurement Data Warehouse stores the monitoring data and the event notifications. One approach to provide the service is with a data warehouse presented in [WIE96]. It will allow collection of the measurement data at distributed, independent sites and will also integrate copies of the data in a common database for querying and analysis. The data warehouse approach allows us to separate the real-time updates of the measurement data from the complex data analysis performed by management applications.

Figure 6 presents information flows from managed nodes to management services. Management agents send predefined event notifications (for example warnings, alarms, etc) to monitoring subsystem and take care of real time queries. The Measurement Data Warehouse is updated separately and can be done by two design methods. Updates to the Measurement Data Warehouse can be done either once a day in a nightly batch run or by sending updates in small cycles.

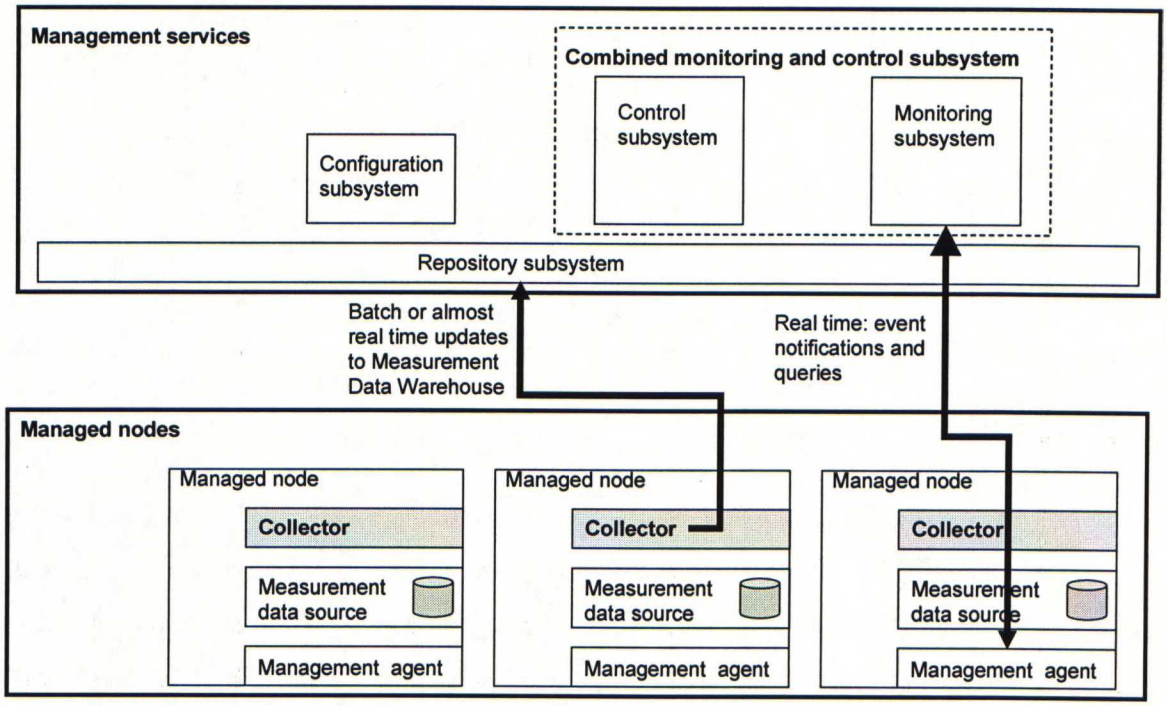

Figure 6: Information flows between management services and managed nodes

The data warehouse approach and implementation options are discussed more detail in chapter 3.

2.1.3.2. Configuration services subsystem.

Structural data that captures the structure and relationships of a system is referred as configuration data. Configuration data has both static and dynamic aspects. Static configuration data describes the organization of the system at start-up. Changes made to the system, while the system is running, create dynamic configuration data. Both a user request to start an application and a process spawning a sub-process represent the types of events that change system's configuration. The configuration services subsystem is responsible for detecting, collecting and maintaining descriptive and location information about entities of the distributed system. Configuration information is stored in the MIR (Management Information Repository).

2.1.3.3. Combined monitoring and control services subsystem.

In the original management framework [BAU94] monitoring and control services were considered as two separate subsystems. Three years later it was proposed that they should be combined into one subsystem implemented through a combination of instrumentation and management agents, with the only difference between control and

monitoring being the direction of information flow: out of a managed process for monitoring and into it for control. Figure 7 visualizes information flows in the subsystem.

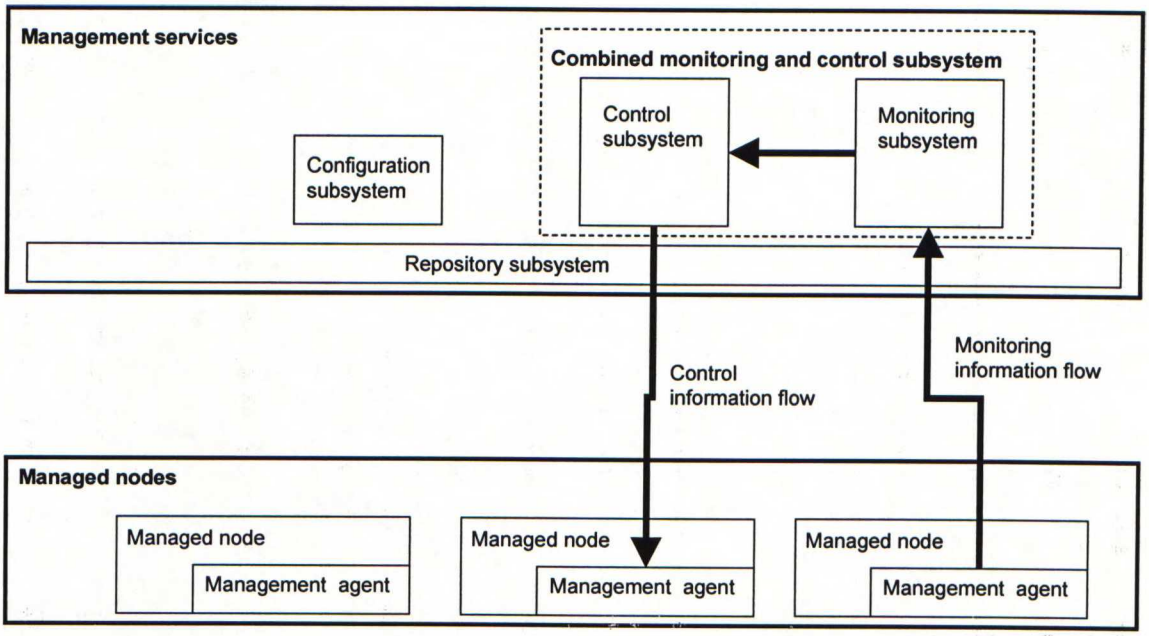

Figure 7: Information flows between control and monitoring subsystems

The monitoring subsystem is responsible for monitoring the behavior of managed objects in the distributed system. It is also responsible for handling real time tasks. Examples of real time tasks are event notifications and real time queries made by management applications or other subsystems.

The control subsystem encompasses a set of components responsible for controlling the behavior of managed objects. Control activities may also be carried out by interacting with management agents. The control subsystem's requests may come from various management applications, from the monitoring subsystem or from the configuration subsystem. For example, the monitoring subsystem or management applications may trigger appropriate control actions to be taken when exceptions on managed objects arise.

2.2. Alternative design approaches and considerations

Although the framework is very generic and easily adaptable it is also originally designed for distributed applications management. In this chapter we try to bring up alternative design approaches and considerations. Motivation to these considerations is that either in some places there is more than one possible way to implement the framework or there are other known solutions that conflict with the framework.

2.2.1. Integration with proprietary management systems

The framework suggests that the management agent should be located in the managed node. Thought this is the best solution it is also worth to notice that in some situations one – probably specially tailored – management agent may also be used to handle more than one managed node. This violates the three-tier structure of the framework, but one can imagine situations where gains are greater than harms.

This kind of exception may be useful in situations where either an object which behavior is wanted to monitor does not support the chosen management protocol (SNMP [CAS90], CMIP [ISO1] or proprietary protocol) or you want to connect a proprietary management system's information flow into the management services. In this case a separate management agent can be used which takes care of the message and information flow routing. In other words a separate management agent can take care of the integration of two separate management systems. Figure 8 illustrates the use of a specially tailored management agent as an integration point to other management systems.

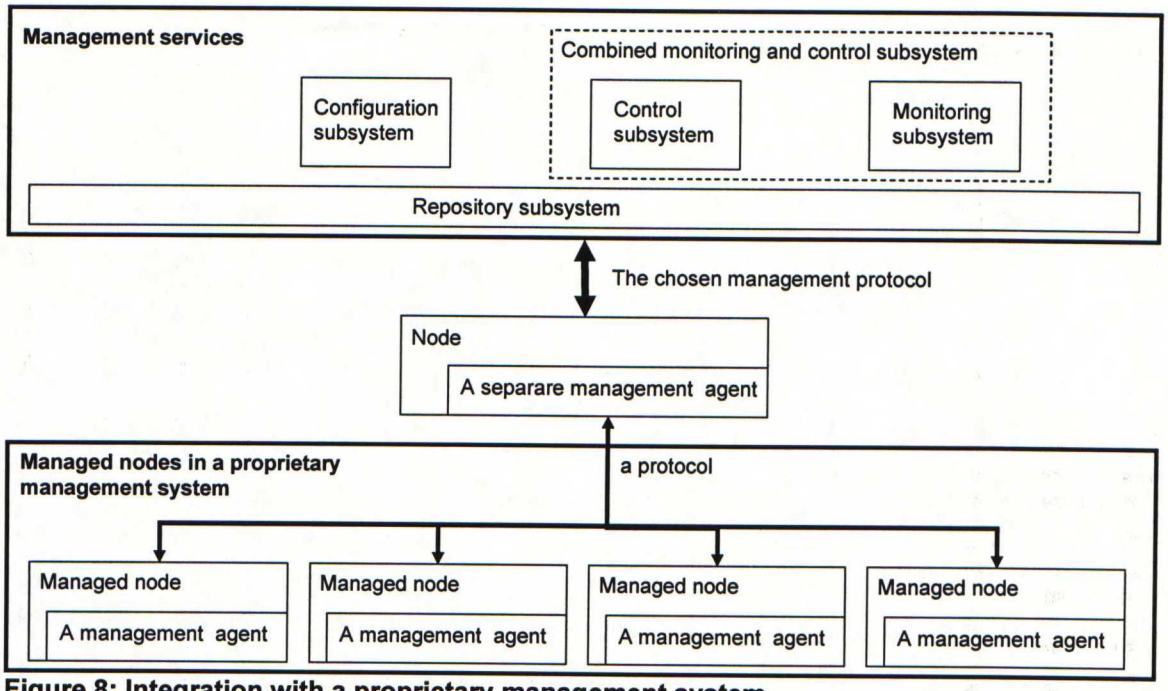

Figure 8: Integration with a proprietary management system

I think it is safe to predict that not all features from a proprietary management system will ever be implemented into the separate management agent. The amount of monitoring data achieved this way may be lower or only the most common control aspects are implemented. This is purely an implementation decision. In some cases proprietary management system's own centralized control interfaces may still be used as a specialist's tools.

Another option is that a proprietary management system is still used for monitoring and control aspects and only historical data is transferred to the repository subsystem. This way management applications can access historical data from a proprietary management system with the same methods it uses for all information retrieval purposes. This is illustrated in Figure 9. A proprietary management system is used in parallel with the framework. The collector (a part of the Measurement Data Warehouse solution) is responsible for propagating data from a local data source into the Measurement Data Warehouse. The data warehouse approach is explained more detail in chapter 3.

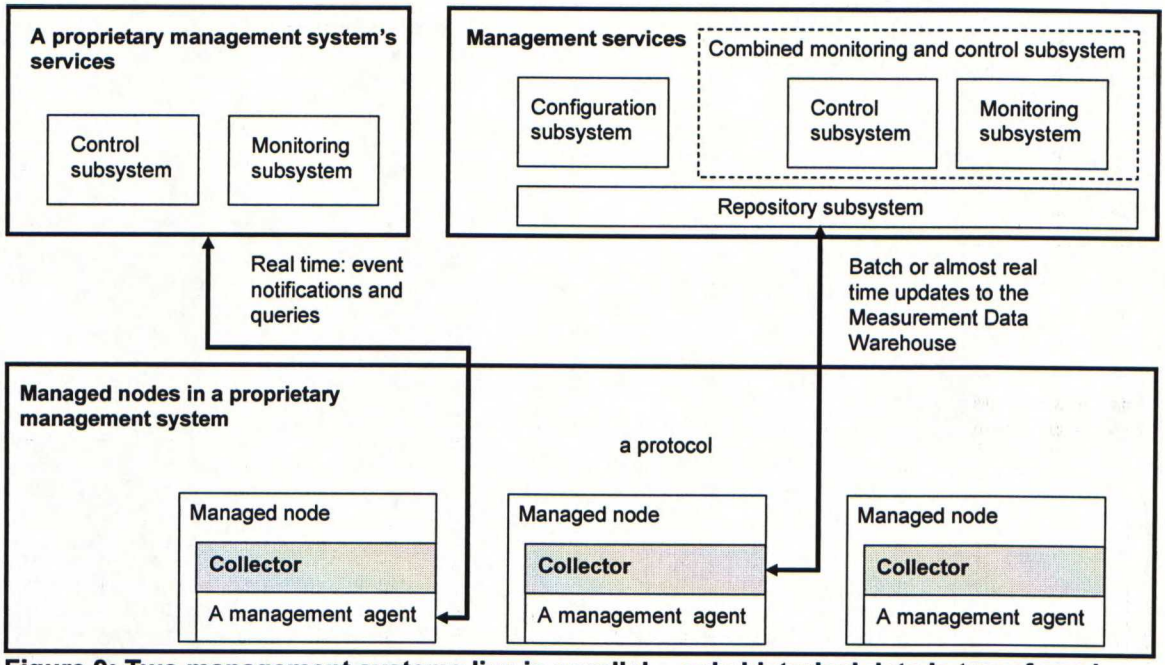

Figure 9: Two management systems live in parallel – only historical data is transferred to the repository subsystem.

This kind of solution violates the framework, but may justify its existence as a temporary solution during the transition period or as a quick and cheap solution if there is reluctance to do major changes.

2.2.2. Data stores – when and where

The framework suggested the data warehouse approach which allows the separation of real-time updates of the measurement data from complex data analysis performed by management applications. Another possibility is to stream all collected data to a centralized historical database immediately [LEE02].

Streaming monitoring data immediately will make management agent's role much lighter. It does not have to make updates to and manage information in the local database. A simple push to a transmission queue will be enough or in case of heavy network load management agents have to maintain a disk buffer where events are pushed first.

In a streaming solution bottleneck points lie in the data transmission path and database updates. Disk buffers are used in critical points to improve scalability. Bottleneck points have been illustrated in Figure 10.

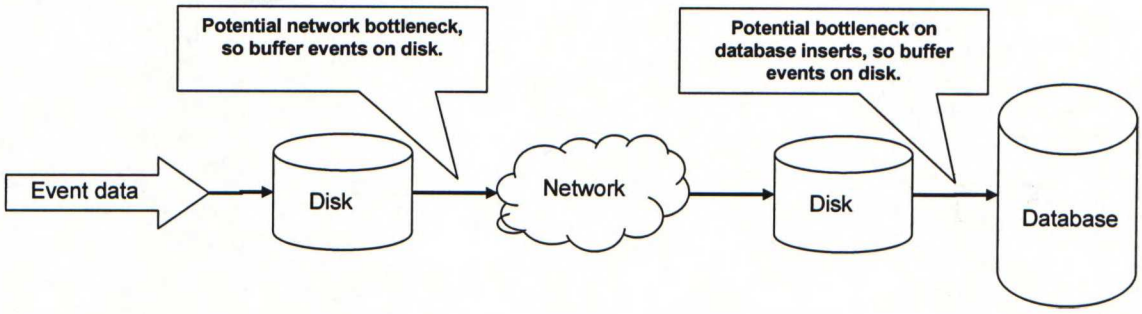

Figure 10: Bottleneck points when streaming monitoring information to a centralized historical database, based on [LEE02].

However in large distributed systems the amount of monitoring data required to identify system's current state (described in detail in chapter 4) will be very high. In a production environment there is a risk that streaming solution will disturb low latency critical applications (for example transaction processing systems). Unfortunately in streaming solution the only way to deal with the problem is to refine parameters of the monitoring data collection. Advantages and disadvantages of different data storing approaches considering enterprise wide environment are evaluated in Table 2.

Data storing approach	Advantages	Disadvantages
Updates performed on databases local to management agents	+ lower usage of network resources + aggregate information enough for most uses + more scalable + performance problems are local	- more complex management agent needed - distributed monitoring of distributed systems
Updates performed directly to centralized historical database	+ simpler management agent + enough information for	- use of network resources - case studies needed for managing enterprise level

	debugging in centralized historical database + real time updates in centralized location	distributed systems

Table 2: Advantages and disadvantages of two different data storing approaches

2.2.3. Separate monitoring and control agents and subsystems

In certain commercial systems separation between monitoring and control subsystems is not so clear. For example in BMC Software's products [BMC2], [BMC3], [BMC4] monitoring subsystem (PATROL) has some control aspects in its management agent as well. The actual control subsystem (CONTROL-M) is a separate product and has its own management agent. Integration between two products is done by a CONTROL-M Integration Module for PATROL (PMI).

The Integration suite enables scheduling and workload decisions based on events generated by the monitoring subsystem. The basic control subsystem supports only scheduling decisions driven by time specifications (combined with other pre-conditions).

In managed nodes this means two management agents in every node. The control agent (CONTROL-M) is a pure control management system. It takes care of scheduling and with the integration suite can also handle event based control. The monitoring agent (PATROL) is a mixture of monitoring and control aspects. It has a complete set of monitoring capabilities but also a limited set of control capabilities focusing mainly on event based control actions. Figure 11 presents the management agents in BMC Software's architecture and how they are connected to each other.

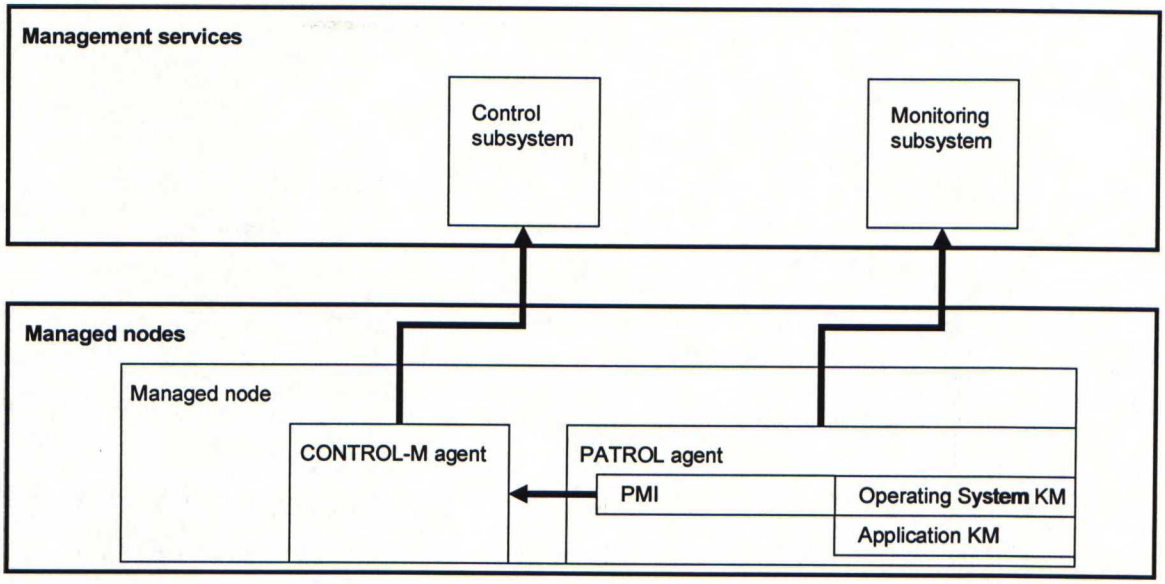

Figure 11: How BMC Software's control and monitoring subsystems are connected.

Reason for this is that they are two separately sold products and in the product point of view there is a demand for limited control aspects in a monitoring product. However from the framework and implementation point of view this kind of architecture is confusing.

Integration between monitoring and control subsystems is done by a sub-module of monitoring agent. The sub-module mediates predefined information from a monitoring agent to a control agent. The control agent then mediates the information to the control subsystem. There is no centralized communication between two subsystems. One can say that there is only distributed integration of two centralized subsystems.

3. THE MEASUREMENT DATA WAREHOUSE (MDW)

After presenting the framework we will now turn our focus on the Measurement Data Warehouse. The Measurement Data Warehouse is a data warehouse solution which in the thesis is proposed as a centralized place where all the measurement information is stored for further analysis. Briefly a data warehouse is a copy of transaction data specifically structured for querying and reporting.

A more complete definition could be. "A data warehouse is a consolidated view of enterprise data - optimized for reporting and analysis. Basically it is an aggregated, sometimes summarized copy of transaction and non-transaction data specifically structured for dynamic queries and reporting. In data warehousing, data and information are extracted from heterogeneous production data sources as they are generated, or in periodic stages, making it simpler and more efficient to run queries over data that originally came from different sources" [WEB4].

Looking from a monitoring data collection point of view we need to collect event notifications and monitoring data to a historical relational database. The data warehouse approach allows us to separate real-time updates of the monitoring data from a complex data analysis performed by management applications. The access to the monitoring data consists mainly of updates by management agents and retrievals by management applications. However there may also be a possibility for applications to return analyzed data back to the data warehouse for further analysis and use for an another management application.

The Measurement Data Warehouse updates are typical database updates in that the new value is independent of the current value, but atypical in that they happen in real time. The real-time characteristic of updates means that the overhead incurred by an update must be minimized and so favors a design in which updates can be performed on databases local to the management agents. In most of the cases retrievals, on the other hand, do not have strict real-time requirements like updates, but they may

involve data from more than one managed object or summaries of the data across one or more dimensions[2].

In next subchapters we will present a general architecture for the data warehouse, propose a simple data model for monitoring and event notification data and consider different possible approaches to make queries more efficient.

3.1. Basic architecture

Figure 12 shows the basic architecture of a warehouse fitted in the framework. Similar kind of basic architecture has been used in Stanford University's data warehouse prototype [WIE96] and in BMC Software's History Loader [BMC5]. Data is collected from each managed node, integrated with data from other sources and stored at the warehouse. Management applications then access data from the warehouse.

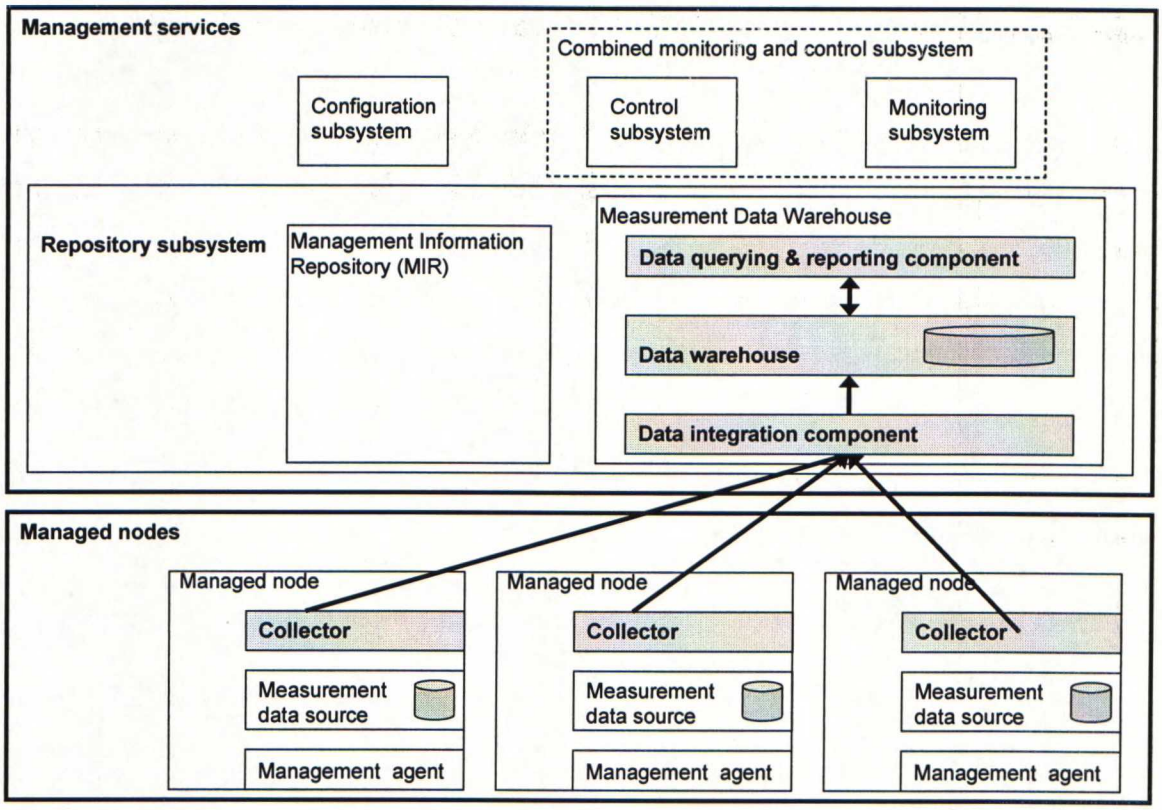

Figure 12: Basic architecture of the Measurement Data Warehouse

[2] In some cases retrievals may have real time requirements. The monitoring subsystem is responsible for handling real time tasks.

There are four main components in the Measurement Data Warehouse system:

- collector
- data integration component
- data warehouse
- data querying and reporting component

The collector is responsible for propagating data from a local measurement data source into the warehouse. The local measurement data source can be of various types, for example a file or an independent database system. The collector takes the data from the source, converts it into repository format and passes the data to the integrator.

The data integration component combines the data from various data sources, translates the data into changes to be applied to views in the warehouse and applies changes to the warehouse data.

The data warehouse is a historical relational database system that maintains the monitoring and event notification data in a form suited to analytical processing. Aggregates along various dimensions may be maintained as materialized views of the base data to support faster querying.

The querying and reporting component is responsible for fulfilling information needs of management applications. Notice that the components are not independent. For example, which views the integration component materializes depends on expected needs of management applications.

In general data warehouse systems consist of two types of tables [MUM97]:

- fact table(s)
- dimension table(s)

A fact table is a table in a data warehouse that contains a tuple of each transaction item. A dimension table is table a in data warehouse which contains information related to a fact table. Data in dimension tables often represents dimensional hierarchies. A dimensional hierarchy is essentially a set of functional dependencies among the attributes of a dimensional table[3].

Considering the measurement data warehouse and the monitoring subsystem, fact tables contain all the monitoring and event notification information extracted from local measurement data sources. Dimension tables contain static support information which helps us to analyze the monitoring data. Structure and design of fact tables and dimension tables will be described more detailed in next subchapter.

3.2. Data model for monitoring information

By definition a data model is the product of the database design process which aims to identify and organize the required data logically and physically. A data model says what information is to be contained in a database, how the information will be used, and how the items in the database will be related to each other. One of the most widely used methods for developing data models is the entity-relationship model (ER) [ULL88].

Also in this thesis data models are presented in ER-format. There is quite a broad selection of data models presented in literature. Models are made for different purposes ranging from simple historical event data models [LEE02] [BMC5] to extensive distributed data management models [RAY97].

A data model for monitoring information has two requirements. First one is, the data model should enable historical relational database for monitoring and event notification data. Second requirement is, the data model should also support management applications needs for data querying and reporting.

---

[3] More formal definition can be found from the website of University of Texas, http://utcdw.utsystem.edu/cdw/PAGES/GLOSSARY/Glossary.htm

Figure 13 presents a simple data model to get monitoring and event notification data into the data warehouse system. First figure presents tables and their relations. Unfortunately because of technical limitations of the A4 paper format the data model is presented in pieces.

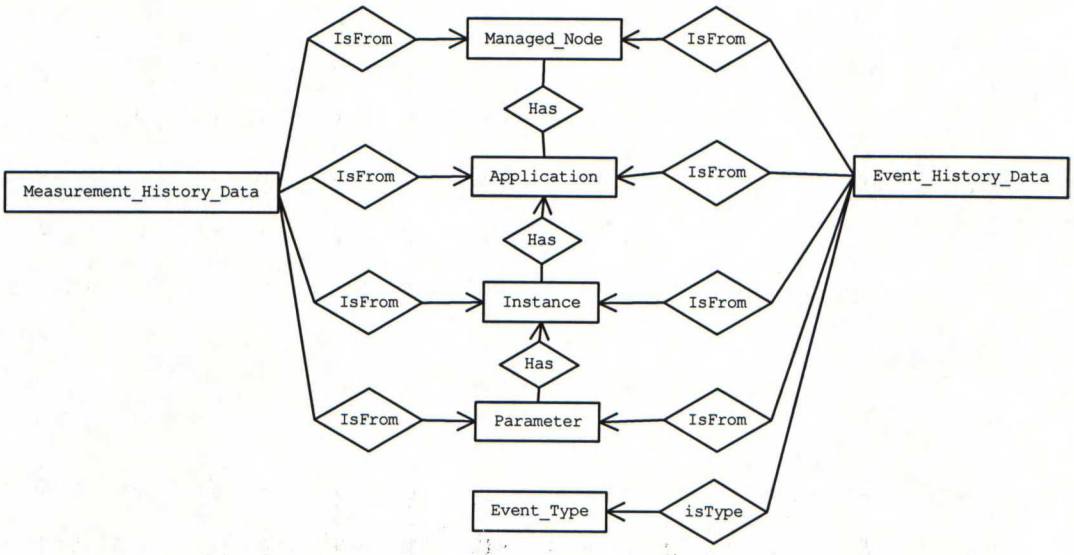

Figure 13: Data model for monitoring information

There are two fact tables: Measurement_History_Data and Event_History_Data. The Measurement_History_Data table contains all the monitoring information. The Event_History_Data table contains event notification information (warnings, alarms, etc) generated by management agent with the help of pre-defined parameters. Fact tables have relationships to Dimension tables.

Dimension tables: Managed_Node, Application, Instance, Parameter and Event_Type provide information related to fact tables. Every managed node has applications. Applications have instances, but instance have to be unambiguous meaning that only one application can have an instance named in a given way (one-to-many relationship). Instances have parameters which also could be called monitored objects. Event_Type specifies the type of event generated (warning, alarm, etc). In some occasions there can be multiple events generated and Event_Type is the only distinctive information.

Figure 14 and Figure 15 present the primary key fields of fact tables. All the dimension tables corresponding fact table has relations have their own field as part of the primary key. Field timestamp is a general way to indicate that there should also be field (or fields) for telling the time and date when a measurement was taken.

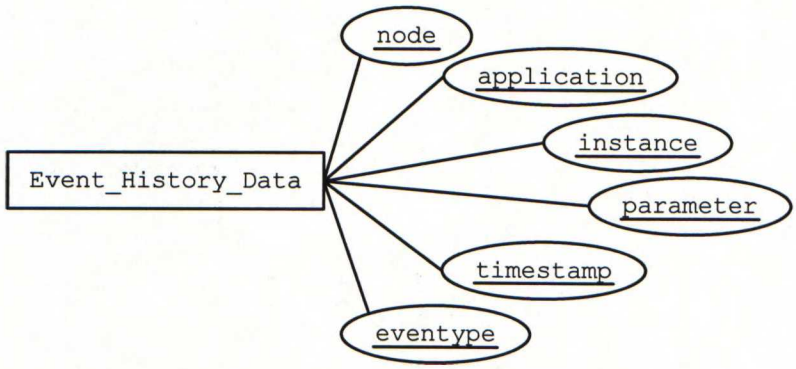

Figure 14: Primary key of the Event_History_Data table

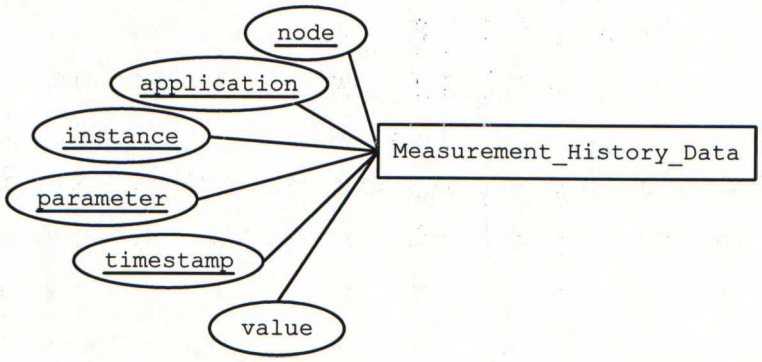

Figure 15: Primary key of the Measurement_History_Data table

Now we have the monitoring information data model that enables historical relation database for monitoring and event notification data. We will present extensions to the data model while we discuss about example management applications in chapter 5.6. The motivation of these extensions is to present possible additions to the data model that make querying and reporting component more useful to management applications.

3.3. Maintaining the information in the warehouse

Without any further development the main information in the Measurement Data Warehouse is in two big fact tables. Data queries of management applications will become extremely heavy if all the information has to be derived from these two tables.

In order to answer aggregate queries quickly, a warehouse will often store a number of summary tables, which are materialized views that aggregate the data in the fact table, possibly after joining it with one or more dimension tables [MUM97]. In this chapter techniques are presented which can be used in selection of a sufficient set of summary tables. Presented techniques are: data cubes and historical summary tables.

3.3.1. Data cubes

The data cube [GRA96] is a convenient way of thinking about multiple aggregate views, all derived from a fact table using different sets of group-by attributes.

Data cubes are popular in OLAP (On-Line Analytical Processing) because they provide an intuitive way for a data analyst to navigate in various levels of summary information in the database. In a data cube, attributes are categorized in dimension attributes on which grouping may be performed and measures which are the results of aggregate functions [MUM97].

Let us take the Measurement_History_Data table as an example. Dimension attributes of the data cube are: node, application, instance and parameter. Measures can be: AVG(value), MIN(value), MAX(value), SUM(value). Figure 16 illustrates a simplified example of the data cube lattice where there are only three dimension attributes: node, application and instance. Example is simplified because the actual lattice of four dimension parameters is quite large.

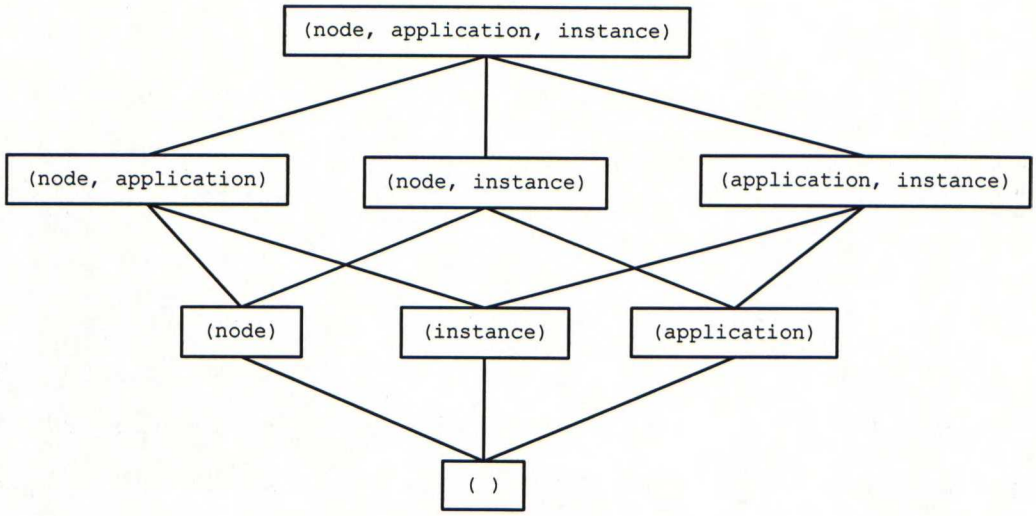

Figure 16: Data cube lattice of simplified example (dimension attributes: node, application and instance)

In figure the point (node, application) represents the cube view corresponding SQL query.

SELECT node, application, AVG(value), MIN(value), MAX(value), SUM(value)
FROM Measurement_History_Data
GROUP BY node, application;

Let us make an assumption that the Measurement_History_Data table contains one month's detailed monitoring information. In above view all monitored values are grouped by so that there is only one row for every node-application pair. And for every row four values are calculated: average, minimum, maximum and sum. Aggregated values come from every node-application-instance-parameter set there is.

Second example is the point (node) and corresponding SQL query is

SELECT node, AVG(value), MIN(value), MAX(value), SUM(value)
FROM Measurement_History_Data
GROUP BY node;

Now there is only one row for every node. And for every row four values are calculated: average, minimum, maximum and sum. And again values come from every existing node-application-instance-parameter set there is.

A data cube with k dimension attributes has 2^k cube views. A cube view means a normal database view which just belongs to a data cube. We have determined four (4) dimension parameters so in this case we would have 16 views. Now we have to ask ourselves: "what have we gained with the data cube?"

$$\text{cube views} = 2^{\text{dimension attributes}} \qquad (1)$$

Considering monitoring data interpretation and the needs of management applications we have gained very little. Monitoring data has to be correlated across time and space, now we have done neither. Views in the data cube group information by dimensional attributes (node, application, instance and parameter). What we would really need is summary tables which summarize monitoring information across time (hour, day, week, month and year) or provide snapshot of all monitored parameters at given time period. This kind of queries will be common from management applications.

The same thing was also realized by I. S. Mumick et al [MUM97]. They stated that in most warehouse and decision support systems, the set of summary tables do not fit into the structure of cube views – they differ in their aggregate functions and the joins they perform with the fact tables.

3.3.2. Historical summary tables

Monitoring and event data stored in original fact tables is in a way raw data. It contains information of single measurements or aggregated information from a short time spans (for example five minute average of a measurement taken every minute). Management applications will frequently make queries where monitoring information is asked to be summarized across time.

The querying and reporting component of the warehouse can of course obtain information with a heavy query to original data. However, a more elegant solution is

to calculate information summarized across time into supplementary fact tables. The fact that at most cases summarized information has to be calculated only once also speaks for this approach.

We have two fact tables in our data model. Let us define five supplementary tables for each fact table. We will take the Measurement_History_Data table as an example:

- Hour_ Measurement_History_Data
- Day_ Measurement_History_Data
- Week_ Measurement_History_Data
- Month_ Measurement_History_Data
- Year_ Measurement_History_Data

Naming convention is time-span_fact-table-name for example Hour_ Measurement_History_Data means that monitoring information is summarized by hour. Supplementary tables ER-model is presented in Figure 17.

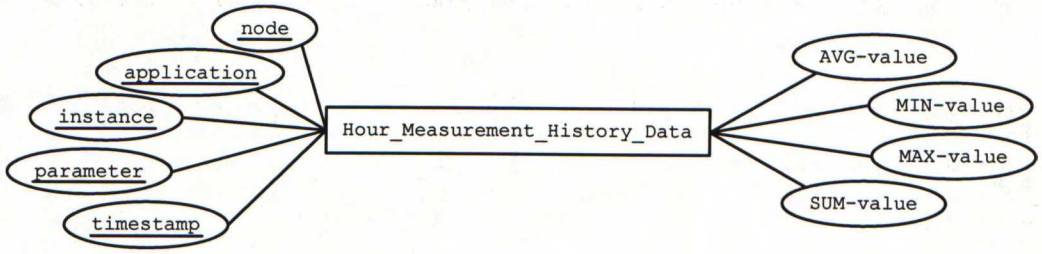

Figure 17: ER-model of summarized measurement history data tables

In supplementary tables there is a row for every node, application, instance, parameter, timestamp combination and summary data of values field include average, minimum, maximum and sum. Supplementary fact tables have same relationships as the table they originate from.

This kind of information is especially useful for management applications like capacity management and service level management. In those there is a constant need to evaluate historical behavior and trends.

4. SCALABILITY AND RESOURCE USAGE CALCULATIONS

After presenting the framework and the Measurement Data Warehouse solution we will now turn our focus on scalability and resource usage calculations. There are two motivations behind this chapter. First is to show that the framework is suitable for enterprise level architecture. Second is to illustrate why it is important to concentrate on minimizing monitoring overhead. Throughout monitoring of a distributed system generates enormous amount of information and it is important to choose a configuration with minimal monitoring overhead.

According to [LEE02], when transferring monitoring data to the centralized data warehouse the most common bottleneck points lay in two locations:

- in data transmission path
- in database updates

In data transmissions, it is important that the solution does not overload the network and thus disturb the functionality of a monitored system. In updates to the data warehouse, it is important to make sure that updates can be done in a reasonable time window. In addition to these parameters we will estimate disk usage needs in two locations:

- the local database in a managed node
- the Measurement_History_Data table in the data warehouse

The local database in a managed node is responsible for storing all the detailed information. It is up to the configuration of a system how long detailed historical information is stored. The Measurement_History_Data table is the place where all the monitoring information of parameters that are transferred to a data warehouse system is initially stored. It is not reasonable (we will shortly show why) to collect historical information to the Measurement Data Warehouse about every parameter monitored by

monitoring subsystem. One has to define a proper set of parameters that are collected and useful to management applications. The set of parameters depends greatly on used management applications.

We will discuss about the need of monitoring information in chapter 5 where example set of management applications is presented. Here we will concentrate on general monitoring information accumulation concerning the Measurement Data Warehouse.

4.1. Amount of monitoring information

Let us make an assumption that in an average managed node there are 20 applications that have 10 instances that have 10 monitored parameters. With this simple generalization we can assume that there are totally 2000 monitored parameters in an average managed node. Figure 18 shows the accumulation of monitoring information from one managed node with two different measurement intervals.

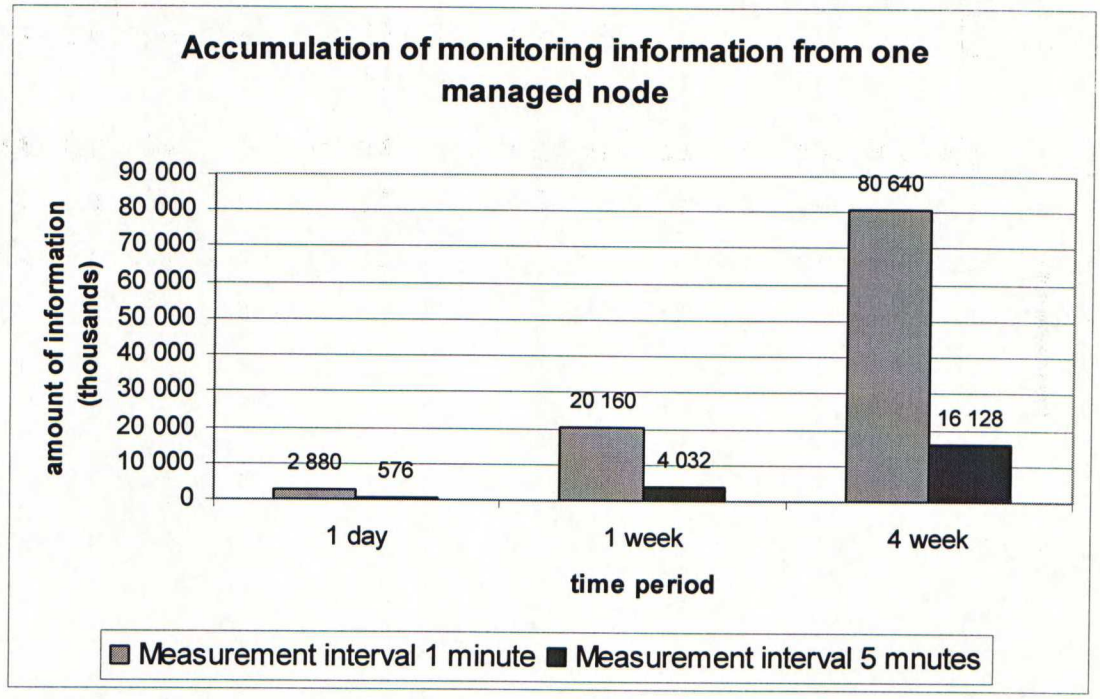

Figure 18: Accumulation of monitoring information from one managed node with two different measurement intervals.

From the Figure 18 we can see that even with five (5) minute time intervals the amount of measurement data in one (1) day will be 576 000 individual measurements.

If you want to keep historical measurement information in the local data source for four (4) weeks, it will mean over 16 million database rows. The amount of data collected to the data warehouse has to be filtered.

Next we will try to estimate the amount of data enterprise level monitoring architecture generates to the Measurement Data Warehouse. To do this we will make assumptions presented in Table 3. Notice that there will be only 30 parameters that are collected to the centralized database from each node.

Parameter	**Small enterprise**	**Medium size enterprise**	**Large enterprise**
Managed nodes	50	200	1000
Collected parameters to centralized database per node	30	30	30
Time interval[4]	5 min	5 min	5 min
Detailed data kept in data warehouse	4 weeks	4 weeks	2 weeks

Table 3: Assumptions made to estimate the amount of data enterprise level monitoring architecture generates.

Results are presented in Figure 19. The amount of data is still very high. Small enterprise would generate 432 000 measurements per day which have to be incorporated to the warehouse in a daily batch run. Also there would be 12 million database rows in the Measurement_History_Data table. Large enterprise would have even more demanding job while dealing with 8 million measurements and over 120 million database rows in Measurement_History_Data table.

[4] Time interval does not necessarily mean that measurements are done in that time interval. Information can also be summarized in collector so that one value moved to the data warehouse is an average of several measurements.

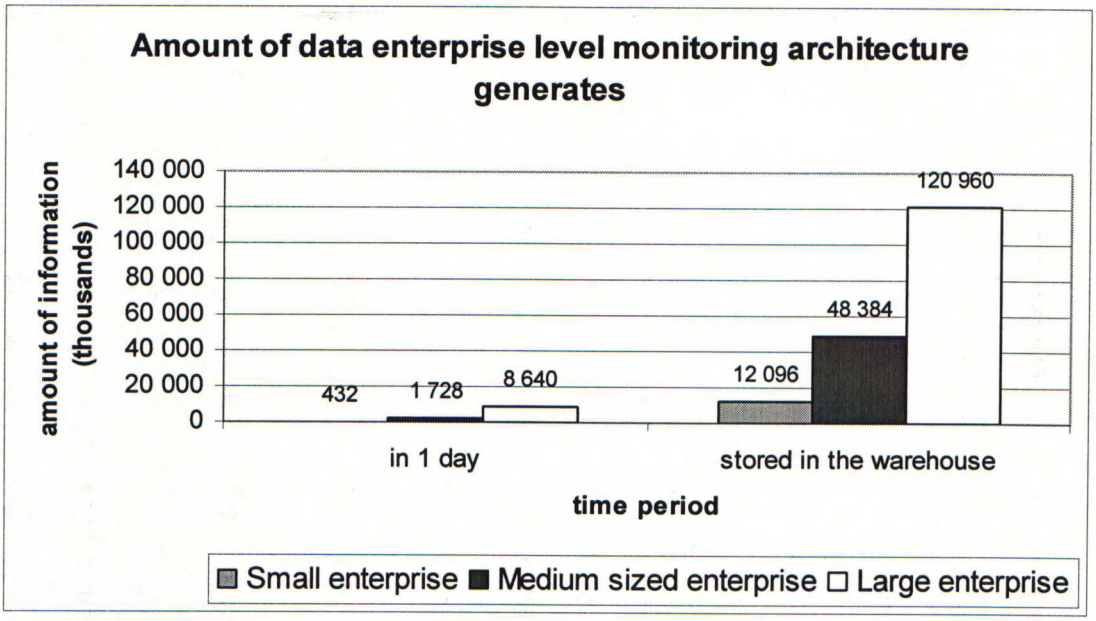

Figure 19: Amount of data enterprise level monitoring architecture generates

From the results above we can learn several lessons:

- The amount of data transferred to the data warehouse has to be chosen carefully.
- Use pre-summarized data when possible.
- Consider carefully how long there is a need to keep detailed data in the data warehouse. Summarized data is enough for most situations and needs.

Next we will adjust our assumptions a bit considering the lessons just learned. Refined assumptions are presented in Table 4. Time interval is adjusted to 15 minutes which is enough for most needs. Notice that this does not mean that measurements are done in that time interval. Information will be summarized in the collector and averaged values will be sent to the data warehouse. Also the storage time of the detailed data is minimized. It is still kept in three days for small and medium sized enterprises but assumption is made that large enterprise will only need one day data.

Parameter	**Small enterprise**	**Medium size enterprise**	**Large enterprise**
Managed nodes	50	200	1000

Collected parameters to centralized database per node	30	30	30
Time interval[5]	15 min	15 min	15 min
Detailed data kept in data warehouse	3 days	3 days	1 day
Disk space need of a database row	125 bytes	125 bytes	125 bytes

Table 4: Refined assumptions to estimate the amount of data enterprise level monitoring architecture generates

Results are presented in Figure 20. Figure 20 shows us that now the amount of data is reasonable. Large enterprise still has to incorporate 2,8 million rows into the data warehouse every day, but it is reasonable.

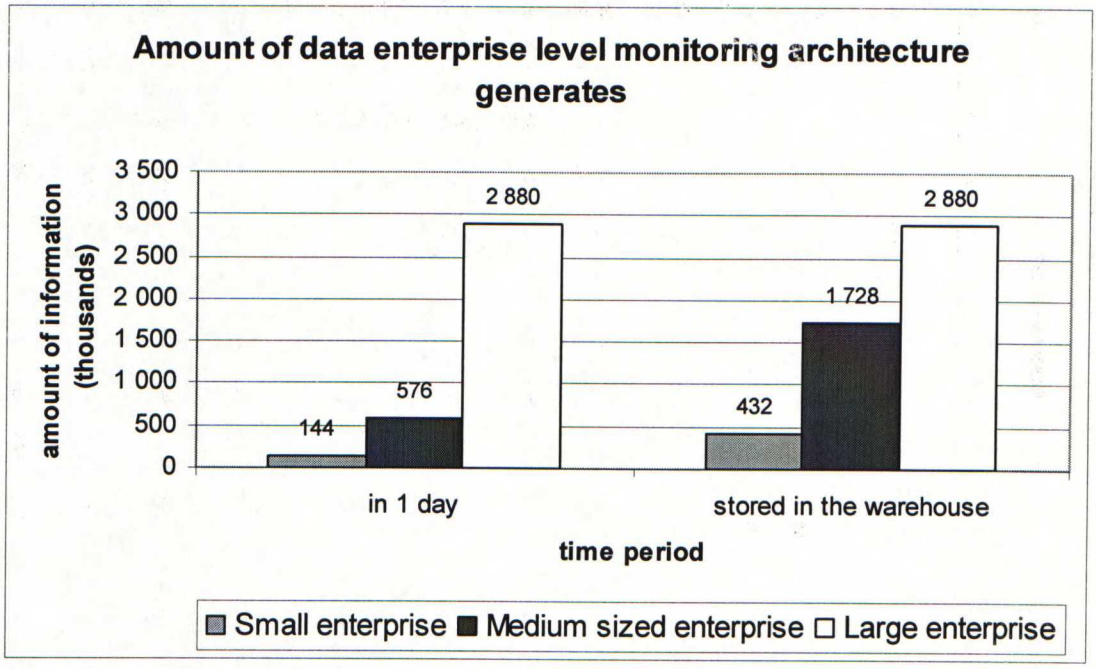

Figure 20: Refined assumptions - amount of data enterprise level monitoring architecture generates

[5] Time interval does not necessarily mean that measurements are done in that time interval. Information can also be summarized in the collector so that one value moved to the data warehouse is an average of several measurements.

4.2. Disk space needs

Next we will try to estimate disk space needs of the local measurement data source maintained by a management agent and the Measurement_History_Data table in the Measurement Data Warehouse.

Storage requirements will vary slightly for each Relational Database Management System. Also the content of a table (for example amount of fields, type and length of fields) determine the exact amount of disk space one database row needs. However according to [BMC5], in general we can estimate that each row of the Measurement_History_Data table will require about 125 bytes.

We will use information from previous chapter (4.1). From Figure 18 we can see the accumulation of monitoring information in one node. From information we can derive estimates to the disk space need of local measurement data source maintained by a management agent. Results are presented in Figure 21.

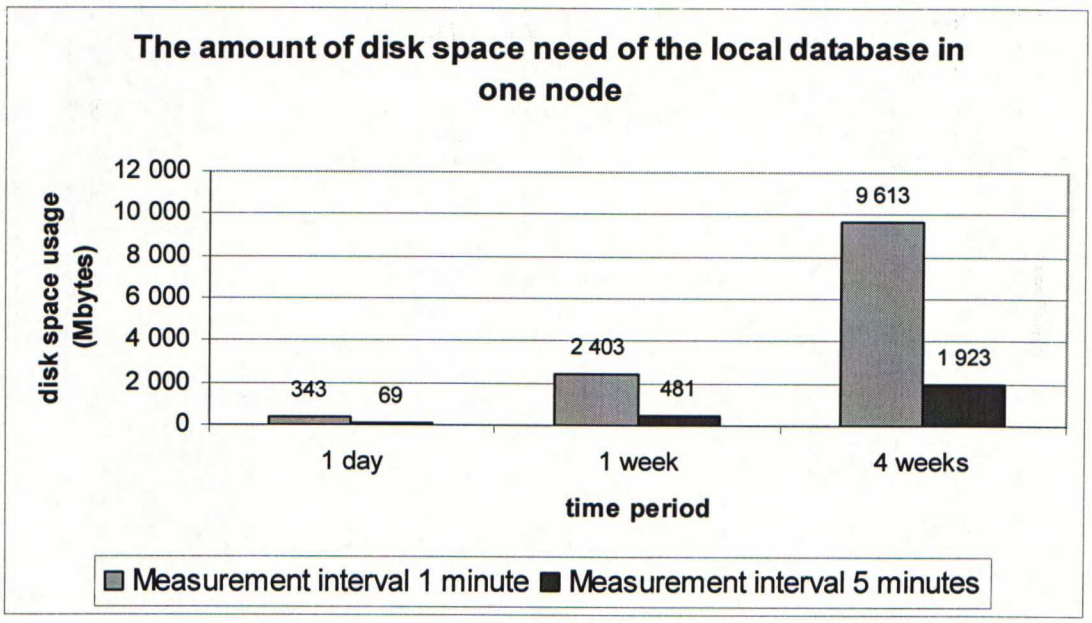

Figure 21: The amount and accumulation of disk space need of the local database in one node

One has to consider carefully how long detailed historical information is stored into the management agent's local database. In one week period there is a need for 480

Mbytes if monitoring interval is five (5) minutes and 2,4 Gbytes if monitoring interval is one (1) minute. In four weeks the disk space need can be anything from 2 to 10 Gbytes.

Next we will turn our focus to Measurement Data Warehouse. From Figure 20 we can see the amount of monitoring data enterprise level architecture generates to the Measurement_History_Data table. From information we can derive estimates of the disk space need of the Measurement_History_Data table. Results are presented in Figure 22.

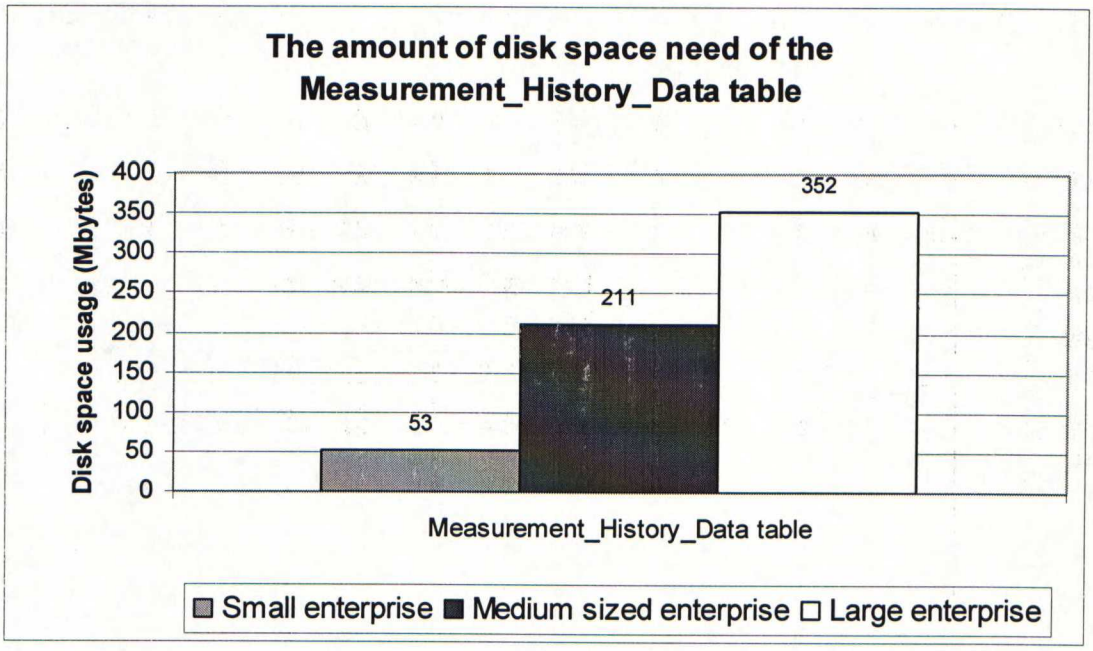

Figure 22: The amount of disk space need of the Measurement_History_Data table

Small enterprise's Measurement_History_Data table needs 53 Mbytes and large enterprise's 352 Mbytes. The amount of disk space need is reasonable. Although we have to remember that we are only talking about Measurement_History_Data table. Supplementary summary tables and views of data cube have their own disk usage needs. Total disk usage needs are harder to estimate cause of the amount of parameters. The disk usage needs of summary tables depend on:

- total amount of parameters collected to the data warehouse

- how long historical data is maintained (for example how long there is a need for hourly or daily summarized parameters)

The disk usage needs of data cube views depend greatly on the chosen implementation strategy and the basic decision whether to implement a complete lattice or not. For example some of the data cube views presented previously demand great deal of disk space.

4.3. Network traffic

According to [LEE02], when transferring monitoring data to the centralized data warehouse the most common bottleneck points lay in two locations: data transmission path and database updates. We have already considered database updates and now it is time to turn our focus on the network. There are two design options how to make data transfers: almost real time and batch. Both have advantages and implementation depends on environment. Almost real time transfer divides the transfers into smaller pieces so network capacity will not be a problem. However if you have to be absolutely sure that nothing interferes with your critical production system a nightly batch transfer is a better option.

As estimated in previous chapter small enterprise's one day measurement information needs 18 Mbytes and large enterprises 352 Mbytes. This is approximately also the amount of data that has to be sent through the network to a centralized location.

Dividing the information to managed nodes this means that every node has to transfer 360 Kbytes of information to a centralized location every day. At the time of the current high speed networks this should not be a problem. In Table 5 are presented the transmission times of a node depending of a slowest link speed in transmission path.

Description	**Slowest link speed in transmission path**	**Transferred data (kbit)**	**Transmission time**
Single line ISDN	64	2 880	45 s
Double line ISDN	128	2 880	23 s

xDSL	256	2 880	11 s
xDSL	512	2 880	6 s
xDSL	1 024	2 880	3 s
xDSL	2 048	2 880	1 s
10 Mbits Ethernet (theoretical max)	5 120	2 880	< 1 s
100 Mbits Ethernet (theoretical max)	51 200	2 880	< 1 s

Table 5: Transmission times from a managed node to a centralized data warehouse depending of the slowest link speed in transmission path (batch transfer)

From Table 5 we can see that the daily batch transfer is not a problem looking from a managed node point of view. Next we will turn our heads to centralized warehouse's side and try to estimate how much traffic it has to be able to receive.

We will make an assumption that the centralized data warehouse system is located in a site where link speeds are fairly high (meaning 10 Mbits Ethernet or higher). Table 6 presents transmission times of a large enterprise (352 Mbytes of data)

Description	**Slowest link speed in transmission path**	**Transferred data (kbit)**	**Transmission time**
10 Mbits Ethernet (theoretical max)	5 120	2 812 500	9 min
100 Mbits Ethernet (theoretical max)	51 200	2 812 500	55 s

Table 6: Large enterprise's transmission times from centralized warehouse's side (batch transfer)

From Table 6 we can see that transmission time of large enterprise's amount of data needs 9 minutes if the link is 10 Mbits Ethernet. From results one can derive following recommendations:

- Measurement Data Warehouse should be located on a site where network transmission capabilities are at least 100 Mbits Ethernet.

- It is a good idea to spread data transfers from different nodes over a longer time period.

5. MANAGEMENT CONCEPTS AND APPLICATIONS

Vast spectrum of management concepts is presented in literature. They address different kind of needs sometimes merely providing a different aspect to same problem or answer to differently formed question. However despite certain overlapping many of them justify their existence by providing a formalized way to solve important problems.

For example, there is some overlapping in capacity management and service level management but their purposes are different. The main difference is that capacity management has a system point of view and service level management customer and service point of view. Monitoring of systems status and performance has an important role in many of these concepts. Thus, it can be defined that monitoring data collection and interpretation has to able to support many management concepts at the same time.

In this chapter we try to clarify different kind of management concepts and group them into logical entities. Example set of management concepts is chosen as a conceptual base of our work. These concepts are presented in more detail. We will discuss monitoring information needs of different management applications. Management applications are tools which aid implementation and use of the chosen management concept within the organization. We will present general high level needs of applications and also try to identify what kind of metrics are relevant to management concepts. In the end we will show how management applications can use repository subsystem's services defined in the framework for their own data storage needs. We will also present examples how the monitoring information data model presented in chapter 3.2 can be extended.

Author's opinion is that management concepts can roughly be divided in two logical groups:

- different system management concepts
- concepts that address more detailed questions or decision making processes

Different system management concepts address IT department's technical need to manage different areas. Following concepts fall into this category:

- network management
- system management
- distributed applications management

Concepts that address more detailed questions or decision making processes are more general processes and try to bind organization's different needs into well defined concepts. Following concepts fall into this category:

- ITIL (IT Infrastructure Library)
- OSI (Open Systems Interconnection) management

Both of these concepts can be further divided into sub-areas and sub-sub-areas. In the thesis we will use different system management concepts and ITIL as a conceptual base for our work. OSI management concepts will be referred to when possible. Let us take Service Level Management, Capacity Management (OSI - performance management) and Incident Management (OSI - fault management) as examples of sub-areas that need monitoring information.

Next we will present different system management concepts and ITIL in more detail. Selected sub-areas are also described

5.1. Different system management concepts

Different system management concepts can support both OSI FCAPS (fault, configuration, accounting, performance and security management) and ITIL's management areas as sub-areas of management. They address IT department's technical need to manage different areas. Concepts are presented in Table 7.

Name of the concept	**Short description**
Network management	Addresses communications

	infrastructure. It covers both PSTN and IP based networks and handles communication lines and network devices.
System management	Addresses systems and services that are scattered in different locations. This means for example devices such as file systems, printers, disk drivers and covers also operating systems and system services.
Distributed application management	Focuses specifically on tools and services for managing the processes and files of applications running within a distributed system. It may be said to sit on top of network management and system management.

Table 7: Different system management concepts

5.2. ITIL (IT Infrastructure Library)

ITIL (IT Infrastructure Library) is a series of documents that are used to aid the implementation of a framework for IT Service Management. Basically it is a collection of industry's best practices and de facto standards that provide a customizable framework that defines how Service Management can be applied within an organization. It was originally developed in the late 1980's by British Government's Central Computer and Telecommunications Agency (CCTA) and is nowadays recognized as the worldwide de facto standard in IT Service Management.

ITIL is really vast concept in and the thesis we will only scratch the surface and try to pinpoint what kind of needs ITIL sets for system monitoring and possibilities what monitoring data collection and interpretation offers for management concepts. ITIL Service Management is divided into two categories: Service Support and Service Delivery. These two categories are further divided into sub-areas.

Service Support is divided into following areas:

- Configuration Management
- Incident Management
- Problem Management
- Change Management
- Service/Help Desk
- Release Management

And Service Delivery into following areas:

- Service Level Management
- Capacity Management
- Continuity Management
- Availability Management
- IT Financial Management

Areas interact with each other and as a whole provide a concept how Service Management can be applied within an organization. Figure 23 gives a quick view of the areas, what they address and their relations.

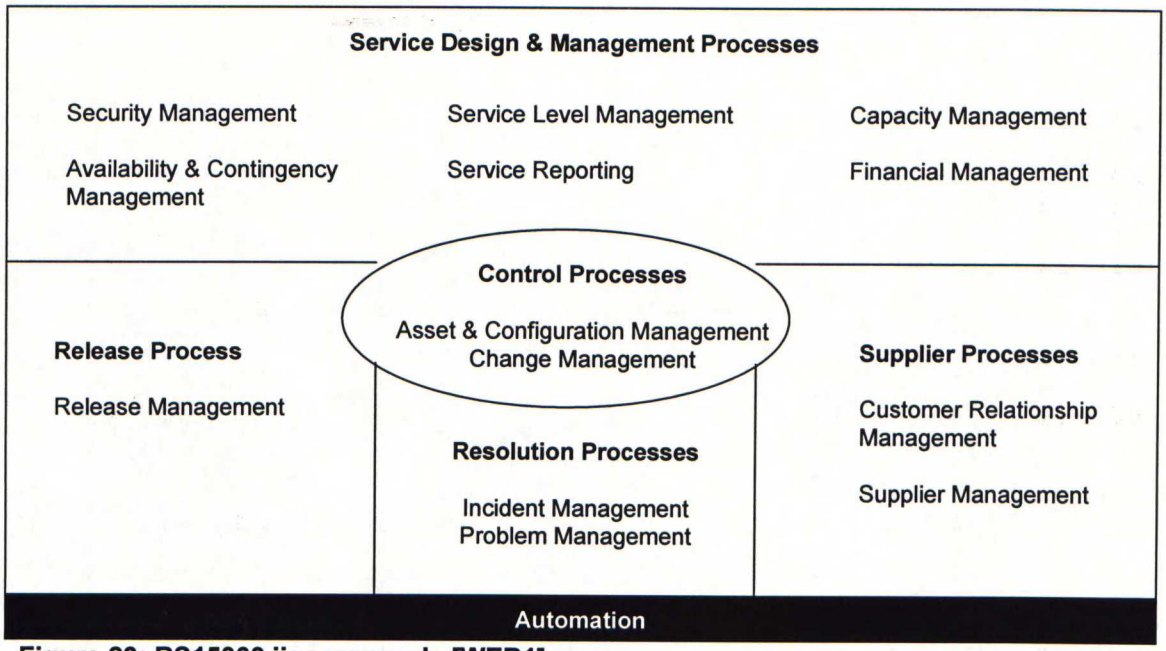

Figure 23: BS15000 jigsaw puzzle [WEB1].

In the thesis we will take: Service Level Management, Capacity Management and Incident Management as examples that have both needs for and possible gains of a properly implemented monitoring data collection and interpretation.

5.3. Service Level Management

Service Level Management (SLM) is responsible for ensuring Service Level Agreements (SLAs) and underpinning Operational Level Agreements (OLAs) are met, and for ensuring that any adverse impact on service quality are kept to a minimum. The process involves assessing the impact of changes upon service quality and SLAs, both when changes are proposed and after they have been implemented [STA1].

For distributed system monitoring point of view the most interesting part are Service Level Agreements and the ability to follow whether the agreed goals come true.

Service-level agreements (SLAs) are contracts between service providers and customers that define the services provided, the metrics associated with these services, acceptable and unacceptable service levels, liabilities on the part of the

service provider and the customer, and actions to be taken in specific circumstances [WEB2].

Although these contracts usually cover the services provided to corporate customers, they can also include the services a company's IT department provides to other business units within the organization. The intra-company SLA is written by the IT department using baseline performance data gathered by various monitoring tools and information gathered from end users about their needs and expectations regarding levels of service. Typically, a separate document is created and maintained for each major application user group, since needs differ from one group to another [MUL99].

From monitoring point of view service level management requires information of SLA metrics agreed with customer. There clearly are two kinds of needs: early warning system and historical data.

Early warning system is responsible for providing warnings when goals either already have been breached or are about to be breached. After a warning appropriate methods can be taken. The early warning system could be implemented as a real time event notification to monitoring subsystem. SLA specific event notifications can be handled like any other system events that need to be investigated. Historical data will be needed for report generation, trend analysis and for simple reason that there is really no idea to define SLA metrics without being able to monitor them.

Monitoring information can also provide itself valuable when SLAs are initially concluded. There is a need of historical performance data which will provide information of current situation and trends.

Of course not all metrics in SLAs are focused on distributed systems. Metrics will also focus on different areas of Service Management. There can for example be metrics for:

- Systems restoral time after incident
- Response times for service requests

- Required contact time of expert services
- Required contact time of basic services

So actually, there is a need for a common measurement database where all measurement information from various sources is stored.

Choosing the right metrics is hard and depends largely on the area of management (network, system, application). Especially in distributed applications management SLA metrics can need considerable time to be defined and metrics depend on the applications special features. However from every area a sample set of metrics can be presented.

One should avoid using absolute boundaries if it is not absolutely necessary. A good example of not so good metric is "all transactions are handled in 2 seconds". From system point of view this is achievable, but would need heavy over provisioning. Normally resource consumption of computer system is highly dynamic and absolute boundaries are hard to meet. Softer boundaries are better for example "90 % of transactions are handled in 2 seconds". This way resources can be dimensioned to meet normal high loads not the peak load.

5.3.1. SLA metrics - network management

Network management addresses communications infrastructure. Meaning for example communication lines, network devices and covers both PSTN (Public Switched Telephone Network) and IP (Internet Protocol) based networks. SLA metrics could address availability and Quality of Service (QoS) parameters. There is a survey [MUL99] of American ISP's and SLAs they provide for corporate customers. Possible metrics are presented in Table 8.

Metric	Short description
Availability	Availability of service, normally measured by percentage. For example 99,9 %. Can also be used to end-to-end

	availability.
MTTR time (Mean Time to Repair)	Mean amount of time before problem is fixed or work-around provided.
Latency (delay)[6]	Maximum end-to-end delay. For example 100 ms.
Successful delivery percentage[6,7]	Can be measured by percentage of packets offered to the network vs. packets that are not successfully transported through the network.

Table 8: Possible SLA metrics of network management

Also there can be different metrics for backbone and access network.

5.3.2. SLA metrics - system management

System management addresses systems and services that are scattered in different locations. This means for example devices such as file systems, printers, disk drivers and covers also operating systems and system services. Some of the performance issues may be handled in Application SLAs. There are a quite a lot of SLA metrics concerning system management that handle "off system" matters. SLA metrics of system management measurable by monitoring could address systems services and general availability. Possible metrics are presented in Table 9.

Metric	Short description
Availability of system services	Availability of system services (for example mail services, virus protection, firewall, domain name service, etc), normally measured by percentage. For

[6] In corporate networks there has to be a clear understanding of bandwidth needs before this can be accomplished

[7] In [MUL99] this is called throughput, but the definition of throughput is "The ammount of data transferred in one direction over a link divided by the time taken to transfer it, usually expressed in bits or bytes per second"

	example 99,9 %.
Availability of nodes	Availability of devices, normally measured by percentage. For example 99,9 %.
MTTR time (Mean Time to Repair)	Mean amount of time before problem is fixed or work-around provided.
MTBF time (Mean Time between Failure)	Mean amount of time between problems with corresponding device or system services.

Table 9: Possible SLA metrics of system management

5.3.3. SLA metrics - distributed applications management

Distributed applications management focuses specifically on tools and services for managing the processes and files of applications running within a distributed system. Especially in distributed applications management it is very difficult to define metrics that would suit for all applications. In here we have divided applications into three generalized groups:

- batch processing systems
- transaction processing systems,
- Web services or client-server applications

Batch processing system is a system which operation is based on heavy calculations and batch runs. It can be a CPU bound calculation application or a mainframe database system. Transaction processing system handles lots of transactions. It can be e.g. a stock exchange system or an electronic commerce system. Web services or client-server applications are systems which functionality is based on queries users make. They can be WWW-services or services which connect to a main database using the client-server architecture.

Distributed application can at the same time belong to several – or even all – of these groups. Possible SLA metrics for each group are presented in Table 10. There are

some considerations presented in [KRI98] concerning the SLA parameters of an electronic commerce web service.

Metric	**Short description**
General	
End-to-end availability	Availability, normally measured by percentage. For example 99,9 %.
Batch processing systems	
Batch turnaround times	Batch turnaround times for business critical runs. Can be defined for example by execution time or by time when end products should be available
OnLine Transaction Processing (OLTP) response times	Response times for queries made on line
Transaction processing systems	
End-to-end transaction times	Time difference from the time when request was made to time when conformation is received
Transactions per second	How many transactions per second system should be able to handle. Can be bound to transaction times. For example 50 transactions per second with maximum response time of two seconds
Total number of transactions	The number of transactions system should be able to handle in given time period.
Web services or client-server applications	
Users connected	The number of users that should be able

	to use system/server concurrently. Can be bound to response times
Response times of static content	Response times of queries concerning static content. For example static html-pages.
Response times of database queries	Response times of dynamic queries. For example database queries of product's current availability.
Response times of transaction requests	Response times of transaction requests.
Response times of logical combinations	Response time of a combination of requests. For example, static query, database query of a product and transaction request concerning the product.
File download times	How long does it take to download a file

Table 10: Possible SLA metrics of distributed applications

5.4. Capacity management

Capacity management (OSI performance management) analyses and controls the usage of system resources. Examples of performance management subjects in different management concepts are:

- network management – QoS parameters (loss, delays, availability)
- system management – resource usage (CPU, memory usage)
- distributed applications management – total throughput (transactions per second)

The most general definition of capacity management is that it is the collection of processes by which the response time and batch service requirements of applications software are consistently and reliably provided, at all times and at a reasonable cost [KOL86].

ITIL's capacity management consists of three sub-areas [STA2] which are presented in Table 11.

Sub-area	Short description
Business Capacity Management	Is responsible for ensuring that the future business requirements are considered. This can be achieved by using existing data of the current resource utilization of the various services to trend and forecast future requirements.
Service Capacity Management	Focuses on the management of the performance of the operational services. It is responsible for ensuring that performance of provided services is within the targets of SLAs. That performance is monitored and measured and that collected data is interpreted.
Resource Capacity Management	Focuses on the management of individual components of the IT infrastructure. It is responsible that components within the infrastructure that have finite resources are monitored and that collected data is interpreted

Table 11: Three sub-areas of capacity management

Each of the sub-areas carries out many of the same activities, but has very different focus. Central part of the capacity management is Capacity Management Database (CMDB) where both measured and forecasted performance data is stored. From monitoring information collection point of view capacity management requires information of system's key parameters and summarized historical monitoring data. Data collected through monitoring components has to be combined with forecast performance data. Again there is a need for a common measurement database where all measurement information from various sources is stored.

Exact measured metrics vary from implementation to implementation but we can give a general selection of metrics that could be collected. There is an old rule in math that could be raised above all "Keep it as simple as possible and not a bit more simple than

that". Too many metrics only cause confusion and you will be blinded by numbers. Carefully choose metrics that has the most value and concentrate on them.

Some of the metrics are the same that are used in service level management. Capacity management can also provide a proactive approach to Service Level Management. Service level agreement (SLA) needs a forecast of system and user growth with input from both sides. Performance management process can be used to find out current trends in systems resource usage and create projections whether current resources are sufficient to handle forecast growth.

Metrics in different general management areas: network, system and distributed applications management also vary. In Table 12 a selection of metrics that could be suitable for different areas are presented.

Performance management in	**Possible performance metrics**
Network management	Link utilization Packet loss (of an interface) Packet loss (network wide) Latency (end-to-end delay) Amount of bandwidth used
Systems management	Processor utilization Disk I/O Swap file usage File system utilization System Paging Memory Usage
Distributed applications management	<u>General:</u> Processor utilization of processes Memory usage of processes Disk I/O usage of processes <u>Transaction processing systems:</u> End-to-end transaction times

	Transactions per second Total number of transaction in given time period Batch processing systems: Batch turnaround times Web services and client server systems: Users connected Response times of - static pages - database queries - transaction requests

Table 12: Possible capacity management metrics for different management areas

5.5. Incident Management

Incident Management (OSI - Fault Management) handles problems reported by customer or management agent responsible for monitoring. Often in a fault there will be many error messages coming from customers and/or from monitoring system. One has to figure out a fault and fix it.

Incident management's responsibility is to detect, isolate and repair abnormal system behavior. There might also be a system which automatically tries to pinpoint the actual fault. From monitoring point of view incident management requires lots of monitored sources (objects) and real time event notifications so that incidents are detected before they become major problems.

Incident management also has its own measurement data collection needs. One has to collect, interpret and display measurement information about the behavior of incident management process itself. This information has to be combined with information from other processes. Again there is a need for a common measurement database where all measurement information from various sources is stored.

5.6. Additions to the data model

The monitoring information data model presented in chapter 3.2 is enough for getting the monitoring data into the data warehouse. Simple additions proposed here are meant to be used by management applications so that they do not have to store all additional information by themselves. Management applications analyze the data and possibly return the data to the warehouse for future analyzes or to be used by other applications.

In this case the Measurement Data Warehouse can be considered as a common measurement database where all measurement information from various sources is stored. This can substantially speed up the management application development process. This kind of approach is illustrated in Figure 24. Of course another possibility is that management applications use their own database solutions for supplementary data storage needs.

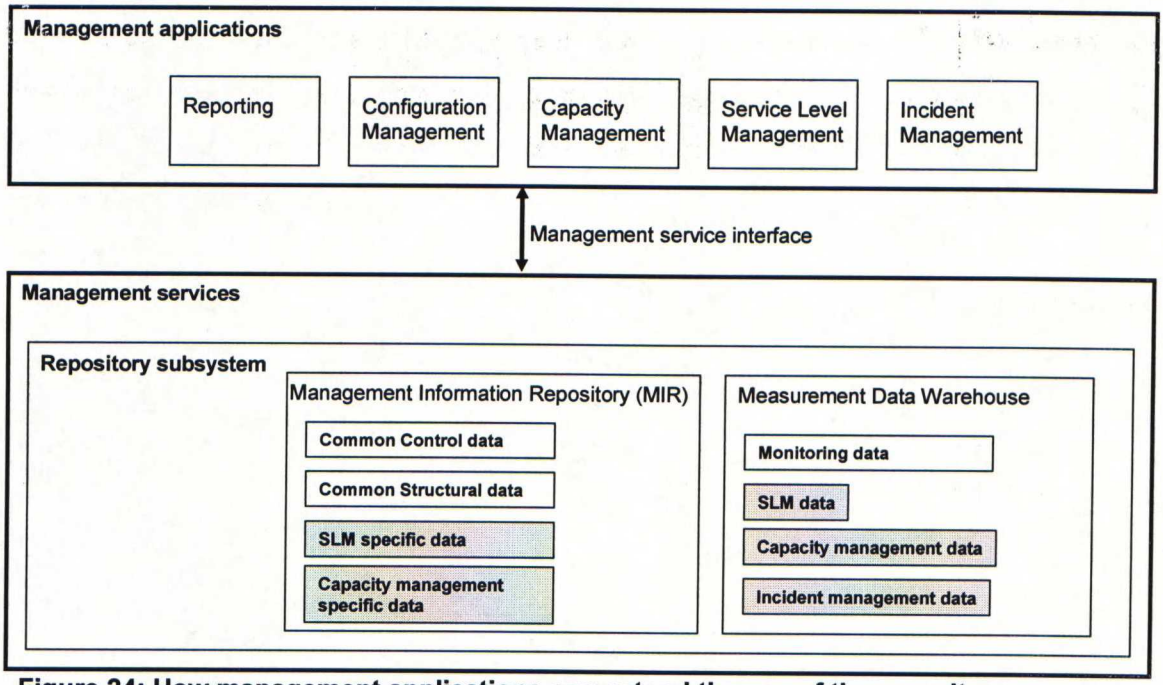

Figure 24: How management applications can extend the use of the repository subsystem

Implementation of additional data models depends greatly of management applications and used systems. In the thesis we will just present general idea how the original data model can be extended.

Additions can be implemented to the Measurement Information Repository and management applications can retrieve information from and maintain the information in the MIR. The Measurement Data Warehouse can provide additional tables for management applications storage needs.

5.6.1. Domain addition

Managed nodes can be grouped into domains. Domains do not have to be actual, this way it is also possible to make logical domains for example all Sun Solaris computers or all network nodes. Domains can also consist of other domains or even a mixture of sub-domains and individual nodes. Figure 25 present the ER-model of the domain addition.

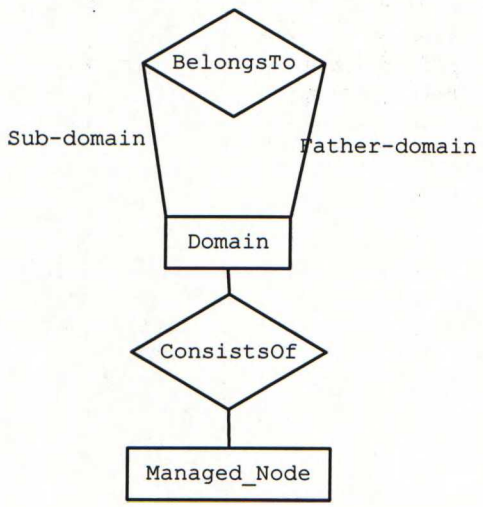

Figure 25: ER-model of the domain addition

The domain can help management applications and maintenance of information. Information of different nodes does not have to be maintained separately by independent applications. While making queries management applications can refer to the domain and get aggregated information from multiple managed nodes.

5.6.2. Management models addition

Management models may also be updated to the MIR and management application can only refer to a model it wants to use and get the information it needs. Figure 26 presents the general ER-model of management model addition.

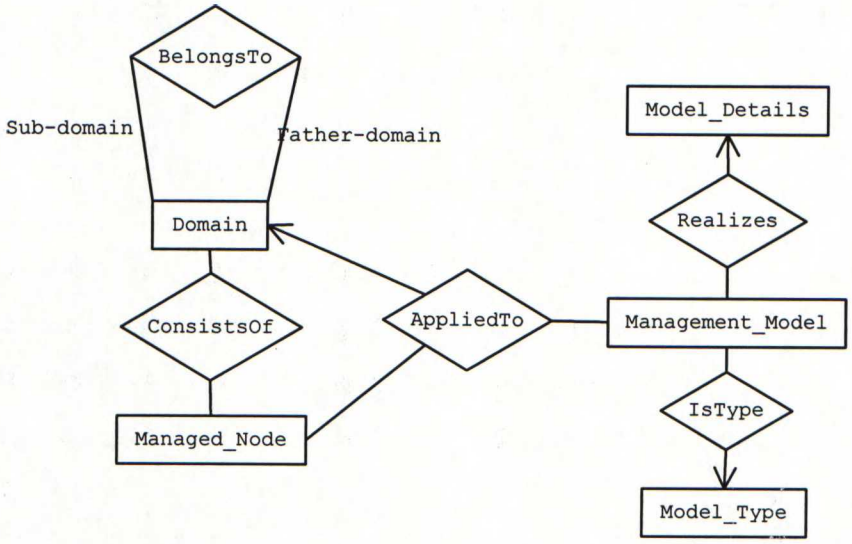

Figure 26: ER-model of the management model addition

We have used a realization where actual details of the management model are hidden into a table Model_Details. It is a generalization and we assume that all the necessary information concerning what model actually does can be obtained from there. This kind on information can for example be queries model has to make, threshold values, future trend projections and where the results are stored. Model can be of one type (for example, service level management model). The management model can be applied to one Domain and also for an additional group of managed nodes.

This way models can be shared with management applications. Practical use could for example be a general reporting application which could use models from many management applications while generating status and performance reports to customers.

5.6.3. Distributed application addition

There may also be a second relationship between managed nodes and applications indicating distributed applications. Managed_Node and Application are familiar from the basic data model. Now we add Distributed_Application. The ER-model is presented in Figure 27.

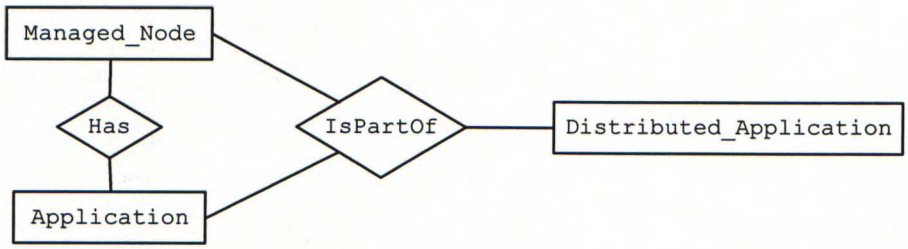

Figure 27: ER-model of the distributed application addition

Management applications may refer to Distributed_Application to obtain all information concerning the behavior of one particular distributed application. Notice that the data model also allows multiple applications of the distributed application to run on the same node.

For example in service level management this kind of information is useful. With one query management applications can obtain all information concerning distributed application's SLA metrics.

6. CONCLUSIONS

In the thesis we have presented the common management framework, considered its suitability for enterprise level monitoring system, introduced the Measurement Data Warehouse for monitoring information storing and shown what kind of gains management concepts can get from a properly implemented monitoring system. The aim has been to give general idea how distributed systems monitoring can be arranged and what are the gains.

Three-tier solution for enterprise level monitoring architecture has been presented. Agents located in every managed node are responsible for collecting the information and maintaining the detailed data in a local data source. The data warehouse solution is responsible for collecting the information to the centralized database. Management applications can access the data from the data warehouse. All this has been encapsulated in the common management framework.

We have presented the Measurement Data Warehouse and shown how monitoring information can be archived in there. We have also presented the monitoring information data model and considered two different techniques to make the data in the warehouse easier and more efficient to access by management applications. Considered techniques have been: data cubes and historical summary tables. We have also shown how management applications can use the data warehouse from their own data storing needs and how the monitoring information data model may be extended.

We have addressed scalability and resource usage issues with theoretical calculations. Aim was to show that the framework is suitable for enterprise level architecture. We have also demonstrated why it is important to choose a configuration with minimal monitoring overhead. Outcome of our results was that no obvious bottleneck point was found.

At the end we have addressed different management concepts and shown what they can gain from a properly implemented monitoring system. We have chosen ITIL's service level management, capacity management and incident management as an

example set of sub-areas that have gains from and needs for monitoring system. We have also discussed metrics which different management concepts may find useful.

Let us now revisit briefly all the main areas of the thesis and sum up our findings and propositions.

6.1. The common framework

Bauer's three-level architecture is very suitable for an enterprise level monitoring system. The framework is presented in Figure 28.

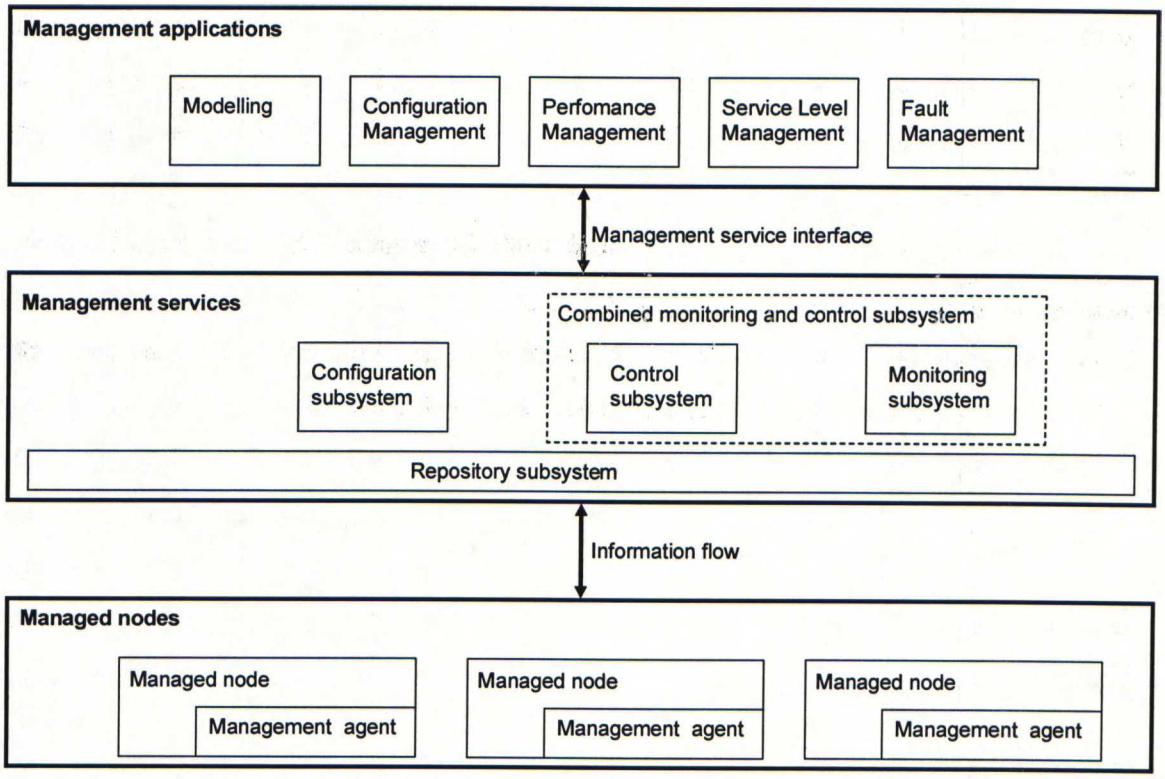

Figure 28: Bauer's distributed applications management framework

Looking from the monitoring system point of view the most interesting components are located in the management services layer. Those components are the repository subsystem and the combined monitoring and control subsystem. The Repository subsystem is divided into two parts: Management Information Repository (MIR) and Measurement Data Warehouse (MDW). Division is done because no single database system could efficiently support three types of data (structural, control and measurement data). Figure 29 presents the repository subsystem in detail.

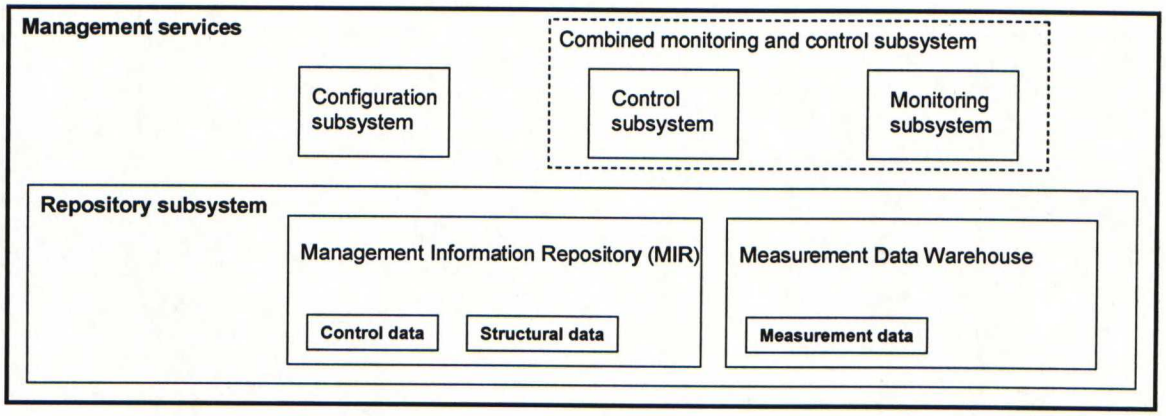

Figure 29: Repository data subsystems in detail

Monitoring information is archived into the Measurement Data Warehouse. But updates are not made in real time and before the transfer the information is filtered and summarized. The amount of data that is generated by monitoring of distributed systems is enormous. Only the important and carefully chosen data is stored to the data warehouse.

The monitoring subsystem is responsible for handling real time event notifications and real time queries. It communicates with management agents and enables monitoring of system's current state. Detailed information is stored locally into the measurement data source maintained by a management agent. Information flows between management services and managed nodes are presented in Figure 30.

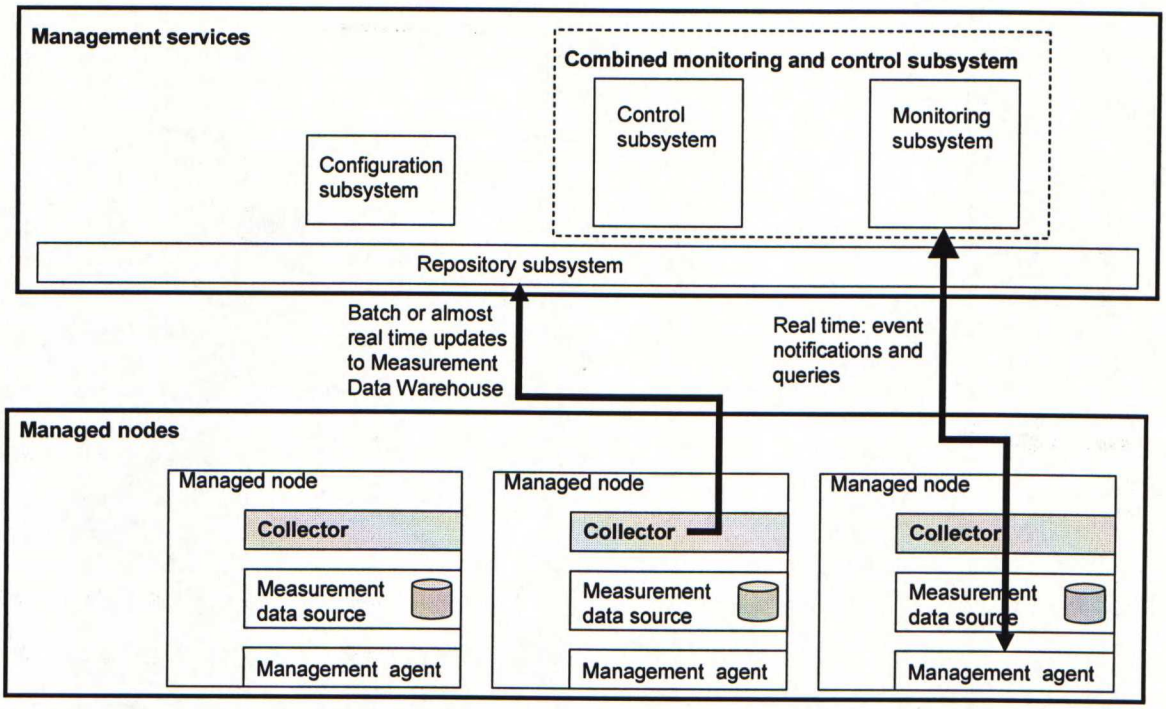

Figure 30: Information flows between management services and managed nodes

6.2. The measurement data warehouse

The data warehouse approach allows us to separate real time updates of monitoring data from the complex data analysis performed by management applications. Detailed information is stored locally into the measurement data source maintained by a management agent. The four main components in the measurement data warehouse system are (Figure 31):

- collector
- data integration component
- data warehouse
- data querying and reporting component

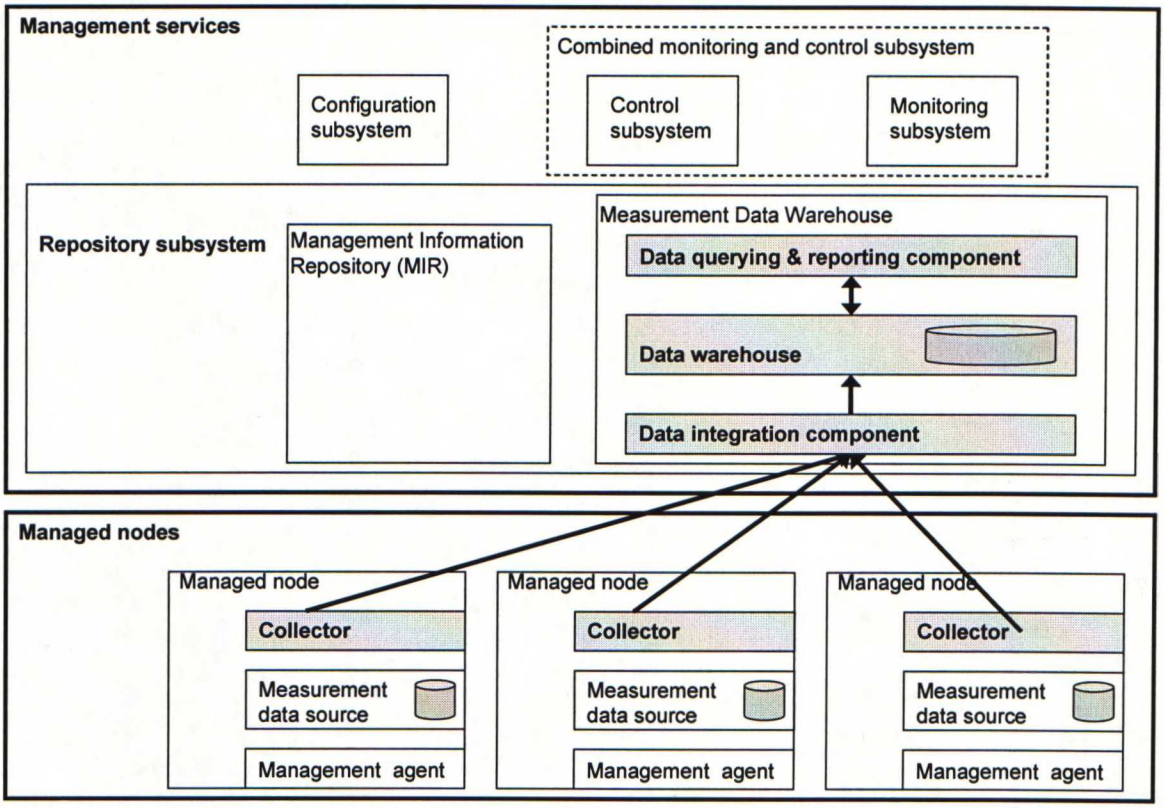

Figure 31: Basic architecture of the Measurement Data Warehouse

The collector is responsible for propagating data from a local measurement data source into the warehouse. The data integration component combines data from the various data sources, translates the data into changes to be applied to the views in the warehouse and applies the changes to the warehouse data. The data warehouse is a historical relational database system that maintains the monitoring and event notification data in a form suited to analytical processing. The querying and reporting component is responsible for fulfilling the information needs of management applications.

We need a data model for monitoring information to get the data to the warehouse. The monitoring information data model is presented in Figure 32.

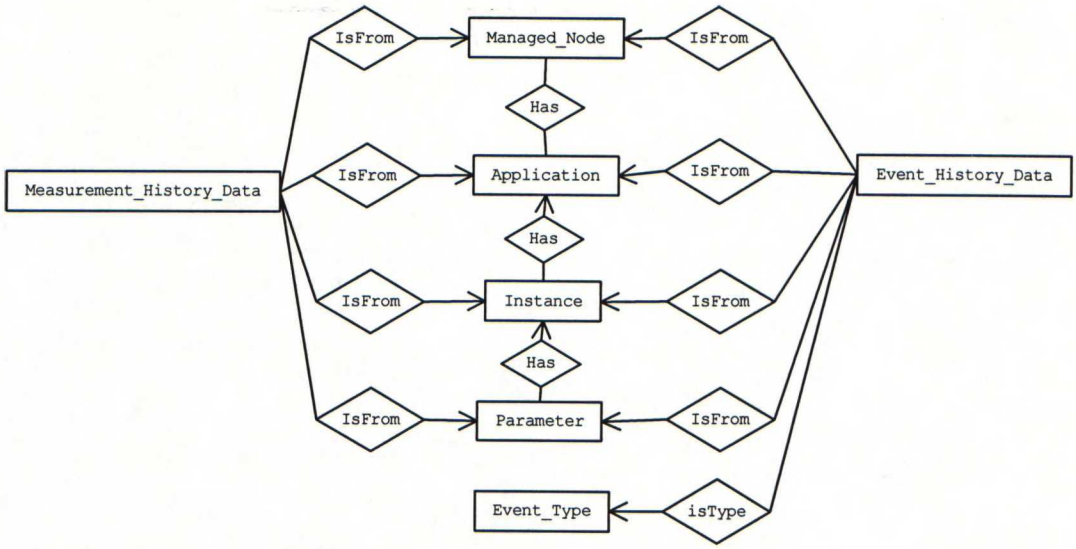

Figure 32: The data model for monitoring information

The data model is enough for getting monitoring information into the data warehouse. In order to answer aggregate queries quickly we also need historical summary tables. Management applications will frequently make queries where monitoring information is asked to be summarized by time. This can be achieved by defining supplementary fact tables. In supplementary fact tables data is summarized by time span (hour, day, week, month or year). This kind of information is especially useful for management applications like capacity management and service level management. On those there is a constant need to evaluate historical behavior and trends.

We have also evaluated the use of data cubes in our work. Findings were that data cubes do not provide any additional value, because they summarize the data in unusable way.

6.3. Scalability and resource usage

In theoretical calculations, we have made estimates of how many resources the proposed monitoring architecture requires in selected key points. Estimates have been made of six factors:

- Amount of monitoring information in a managed node.

- Amount of monitoring information in the Measurement_History_Data table in the data warehouse.
- Disk space need of the local measurement data source in a managed node.
- Disk space need of the Measurement_History_Data table in the data warehouse.
- Network traffic from a managed node to the centralized location.
- Network traffic looking from centralized location's point of view.

Resource consumption has been estimated for three imaginary enterprises: small, medium sized and large enterprise. Outcome of our results was that no obvious bottleneck point was found. However, one should treat results with certain precaution. Calculations made were theoretical and in no way complete. However, we are able to form several points of observation from the results. Points of observation are presented in Table 13.

Focus	**Points of observation**
Managed node	• Consider carefully how long there is a need to keep detailed historical information in the local measurement data source.
Network	• Measurement Data Warehouse should be located in a site where network transmission capabilities are at least 100 Mbits Ethernet. • Spread data transfers from different managed nodes to a longer time period.
Centralized data warehouse	• The amount of data transferred to the data warehouse has to be chosen carefully. • Use pre-summarized data when possible. • Consider carefully how long there is a need to keep detailed data in the warehouse.

Table 13: Points of observation based on resource usage calculations

6.4. Management concepts and applications

We have addressed different management concepts and shown what they can gain from a properly implemented monitoring system. Table 14 presents the key benefits per management concept.

Management concept	Gains of properly implemented monitoring system
Service level management	• When SLAs are initially solved there is a need of historical performance data and trends. • Early warning system which tells when SLA goals are about to be breached or have already been breached. • Historical performance data of SLA metrics.
Capacity management	• Historical performance data of systems key parameters.
Incident management	• Systems real time status monitoring can provide a proactive way to deal with incidents.

Table 14: Key benefits of properly implemented monitoring system per management concept

We have also been able to identify a sample set of metrics which management concepts may find valuable. These are presented in tables:

- Table 8: Possible SLA metrics of network management
- Table 9: Possible SLA metrics of system management
- Table 10: Possible SLA metrics of distributed applications
- Table 12: Possible capacity management metrics for different management areas

We have also briefly shown how the Measurement Data Warehouse can also be used as a common measurement database where all measurement information from various sources is stored. This can substantially speed up the management application development process. This kind of approach is illustrated in Figure 33. With the same technique management applications can extend the monitoring information data

model by adding their own relations. Additions can be implemented to the Management Information Repository.

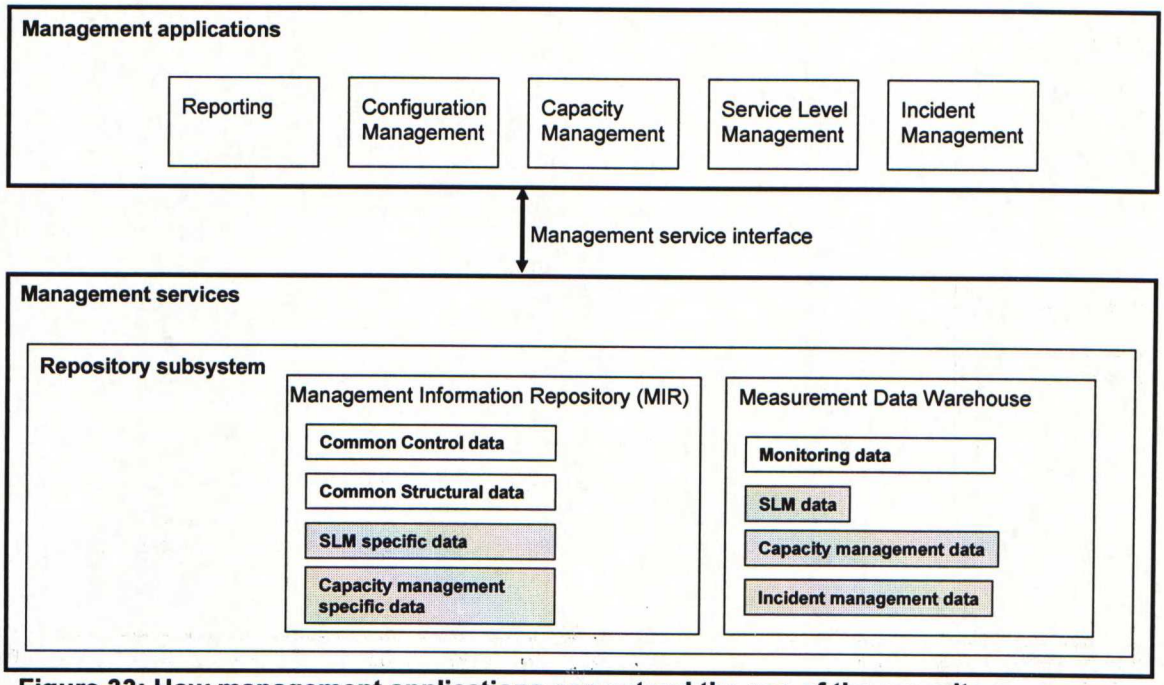

Figure 33: How management applications can extend the use of the repository subsystem

6.5. Discussion and further studies

The aim of the thesis has been to give the general idea how distributed systems monitoring can be arranged and what are the gains. Because of the wide subject there has been only little room for details. In many cases we have restricted ourselves merely to presenting the ideas gathered from many sources and tried to bind them together. There are many open questions that need more theoretical work, working prototypes and case studies.

The framework itself has been prototyped, but it was originally made for distributed applications management. There is a need for a more detailed study which could concentrate on working solutions (commercial, in-house solutions and academic). It would be interesting to see how the framework differs from them. What we know is that a three-tier solution is rather common and some commercial systems (for example BMC Software's PATROL) architecture resembles the framework.

Data warehouses are getting popular and because of this there are quite a lot of papers addressing them. A study concentrating on monitoring information and data warehouse solutions would also be useful. In this study we have very briefly mentioned a couple of solutions to make queries more efficient.

From resource usage and scalability point of view there is a need for prototypes and case studies. In this study we could not find any obvious bottleneck point. This does not mean that there are none. A case study considering monitoring systems fine tuning would also be interesting. Fine tuning means here optimizing the amount of information transferred to a centralized database, how updates are timed and how we can minimize the time window needed for a nightly batch run.

Management concepts and how they are implemented in different companies is a world of its own. Literature gives only guidance and many good advices – actual implementation varies from company to company.

7. REFERENCES

[ABD96] H. Abdu, H. Lutyya and M. A. Bauer. Monitoring Overhead in Distributed Systems: Visualization and Estimation Techniques. Proceedings of the 1996 conference of the Centre for Advanced Studies on Collaborative research. 1996. http://www.cs.ubc.ca/local/reading/proceedings/cascon96/htm/francais/abs/abdu.htm.

[BAU94] M. A. Bauer, P. J. Finnigan, J. W. Hong, J. A. Rolia, T. J. Teorey, and G. A. Winters. Reference Architecture for Distributed Systems Management. IBM Systems Journal 33(3), pp. 426 – 444. 1994. http://www.research.ibm.com/journal/sj/333/ibmsj3303E.pdf.

[BAU97] M. Bauer, R. Bunt. A. El Rayess, P. Finnigan, T. Kunz, H. Lutfiyya, A. Marshall, P. Martin, G. Oster, W. Powley, J. Rolia, D. Taylor and M. Woodside. Services Supporting Management of Distributed Applications and Systems. IBM Systems Journal 36(4), pp. 508 - 526, 1997. http://citeseer.nj.nec.com/bauer97services.html.

[BMC1] BMC Software, http://www.bmc.com/.

[BMC2] BMC Software. CONTROL-M/EM User Guide for version 6.1.02. http://www.bmc.com/supportu/documents/99/57/19957/19957.pdf.

[BMC3] BMC Software. CONTROL-M Integration Module for PATROL User Guide. http://www.bmc.com/supportu/documents/64/59/16459/16459.pdf.

[BMC4] BMC Software. PATROL Console for Unix User Guide. http://www.bmc.com/supportu/documents/76/13/17613/17613.pdf.

[BMC5] BMC Software. PATROL® History Loader User Guide, version 1.4. http://www.bmc.com/supportu/documents/32/36/13236/13236.pdf.

[CAS90] J. D. Case, M. Fedor, M. L. Schoffstall, and C. Devin. Simple Network Management Protocol (SNMP). The Internet Engineering Task Force, Request for Comments 1157. 1990.

[DIN01] P. Dinda and B. Plale. A Unified Relational Approach to Grid Information Services. Grid Forum Working Draft GWDGIS -012-1. 2001. http://www.gridforum.org, http://citeseer.nj.nec.com/dinda01unified.html.

[FIS01] S. Fisher. Relational Model for Information and Monitoring. Technical report, GGF. 2001. http://citeseer.nj.nec.com/fisher01relational.html.

[GRA96] J. Gray, A Bosworth, A. Layman and H. Piresh. Data cube: A Relational Aggregation Operator Generalizing Group-by, Cross-tab, and Sub-totals. Proceedings of the Twelfth IEEE International Conference on Data Engineering, New Orleans, pp. 152-159. 1996. http://citeseer.nj.nec.com/gray97data.html.

[HP1] Hewlett-Packard Company, HP OpenView VantagePoint Operations for UNIX Concepts Guide. http://www.hp.com/.

[ISO1] ISO 9596/ITU-T Rec. X.711. Common Management Information Protocol Specification. 1991.

[JOY87] J. Joyce, G. Lomow, K. Slind and B. Unger. Monitoring distributed systems. ACM Transactions on Computer Systems 5(2), pp. 121 – 150. 1987.

[KOL86] K. W. Kolence. An Overview of the Capacity Management Process. Proceedings of 1986 fall joint computer conference. 1986. ISBN:0-8186-4743-4.

[KRI98] D. Krishnamurthy and J. Rolia. Predicting the QoS of an Electronic Commerce Server: Those Mean Percentiles. ACM SIGMETRICS Performance Evaluation Review, Vol 26, pp. 16 - 22. 1998. http://citeseer.nj.nec.com/98631.html.

[LEE02] J. Lee, D. Gunter, M. Stoufer, and B. Tierney. Monitoring Data Archives for Grid Environments. Proceeding of IEEE Supercomputing. 2002. http://citeseer.nj.nec.com/534091.html.

[MUL99] N. J. Muller. Managing Service Level Agreements. International Journal of Network Management, Vol. 9, pp. 155 – 166. 1999.

[MUM97] Mumick, D. Quass, B. Mumick. Maintenance of Data Cubes and Summary Tables in a Warehouse. Proceedings of the ACM SIGMOD Conference, Tuscon, Arizona. 1997. http://www-db.stanford.edu/warehousing/publications.html.

[RAY97] A. E. Rayess, J.A. Rolia. Automatic Generation of Performance Models Using the Distributed Management Framework (DMF). Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research. 1997. http://www.cs.ubc.ca/local/reading/proceedings/cascon97/cascon97/htm/english/abs/elrayess.htm.

[REN03] R. van Renesse, K. P. Binnan and W. Vogels. Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining. ACM Transactions on Computer Systems 21(2), pp. 164 - 206. 2003.

[SNO88] R. Snodgrass. A Relational Approach to Monitoring Complex Systems. ACM Transactions on Computer Systems 6(2), pp. 157 - 196. 1988.

[SQL92] Database Language SQL. ANSI X3.135-1992

[STA1] The Stationary Office. ITIL Service Support CD, ISBN 0-11-33-00158

[STA2] The Stationary Office. ITIL Service Delivery. ISBN 0-11-33-00174.

[ULL88] J.D. Ullman. Principles of Database and Knowledge-Base Systems. Volume I, Computer Science Press, p. 32. 1988.

[WEB1] BetterManagement.com. http://www.bettermanagement.com/Library/Library.aspx?a=486&LibraryID=5813.

[WEB2] IEC, International Engineering Consortium. www.iec.org/online/tutorials/service_level/

[WEB3] The Open Group Portal to the World of DCE. http://www.opengroup.org/dce/.

[WEB4] Information Builder's website. http://www.informationbuilders.com/definition/data-warehousing.html.

[WIE96] J. L. Wiener, H. Gupta, W. J. Labio, Y. Zhuge, H. Garcia-Molina, and J. Widom. A System Prototype for Warehouse View Maintenance. Proceedings of the Workshop on Materialized Views: Techniques and Applications, Montreal, Canada, pp. 26 - 33. 1996. http://citeseer.nj.nec.com/wiener96system.html.